Communication

Local Instrument Network (LIN) Products

invensys
EUROTHERM

Modbus/Profibus Communications Manual

About

Title	Modbus/Profibus Comms Manual
Part Number	HA 028 014
Issue (Date)	4 (05/2007)

Contents

COMMUNICATIONS MANUAL

CHAPTER 1 MODBUS GATEWAY FACILITY 1-1
1.1 OVERVIEW OF MODBUS GATEWAY FACILITY 1-1
1.1.1 Main features 1-2
1.1.2 Functional description 1-2
1.1.3 Function codes 1-3
1.1.4 Transparent Modbus Access (TMA or TalkThru) 1-3
1.2 PRINCIPLES OF OPERATION 1-4
1.2.1 Operating mode 1-5
1.2.2 Refresh rates and timing information 1-6
1.2.3 Memory use and requirements 1-7
1.2.4 Data conversion 1-8
1.3 USING THE DIAGNOSTIC TABLE 1-10
1.3.1 Internal diagnostic registers 1-10
1.3.2 Modbus table status and control registers 1-10
1.3.3 Diagnostic table registers 1-11
1.4 DIAGNOSTIC FUNCTION CODES 1-12
1.5 MODBUS EXCEPTION RESPONSES 1-12
1.6 NOTES ON MODBUS IMPLEMENTATION 1-13
1.6.1 Modbus (AEG-MODICON) implementation 1-13
1.6.2 JBUS implementation 1-13
1.6.3 Other product implementations 1-13

CHAPTER 2 MODBUS DCM 2-1
2.1 INTRODUCTION 2-1
2.2 INSTRUMENT CONFIGURATION 2-2
2.2.1 Instrument Properties 2-2
2.2.2 Mapping Files 2-2
2.3 THE .UYM FILE 2-3
2.3.1 Scaling 2-5
2.3.2 Commenting 2-5

CHAPTER 3 PROFIBUS 3-1
3.1 INTRODUCTION 3-1
3.2 INSTRUMENT CONFIGURATION 3-2
3.2.1 Instrument Properties 3-2
3.2.2 Mapping Files 3-2
3.3 DEVOLVED CONTROL MODULES (DCMs) 3-3
3.4 THE .UYP FILE 3-4
3.4.1 Introduction 3-4
3.4.2 Commenting 3-9
3.5 INSTALLATION 3-10
3.5.1 Guidelines 3-10
3.5.2 Cubicle wiring 3-10
3.5.3 External profibus networks 3-12
3.5.4 Adding a LIN product to the network 3-14
3.6 TROUBLESHOOTING 3-15
3.7 GLOBAL COMMANDS 3-16
3.8 OPERATION 3-16
3.8.1 I/O data transfer limits 3-16
3.8.2 Data format 3-16
3.9 THE .GSD FILE 3-17
3.9.1 Demand data 3-17
3.10 REDUNDANT (DUPLEX) MODE 3-18
3.10.1 Redundancy decisions 3-18

APPENDIX A CONFIGURATION FILES A-1
A.1 THE _SYSTEM.OPT FILE A-2
A.2 THE _SYSTEM.UXM FILE A-5
A.3 THE NETWORK.UNH FILE A-7

INDEX ..... INDEX-I

CHAPTER 1 MODBUS GATEWAY FACILITY

This section describes the implementation of the Modbus Gateway facility as a part of a LIN instrument, in the following subsections:

- Overview of the Modbus Gateway Facility (Section 1.1)
- Principles of operation (Section 1.2)
- Using the diagnostic table (Section 1.3)
- Modbus diagnostic function codes (Section 1.4)
- Modbus exception responses (Section 1.5)
- Notes on Modbus implementation (Section 1.6)

Note Some LIN products using a Modbus Gateway do not support Modbus Master mode.

1.1 OVERVIEW OF MODBUS GATEWAY FACILITY

The Modbus Gateway facility provides a Modbus interface to the LIN Database via the Serial or Modbus TCP communications interface type on either a Modbus Master or Modbus Slave device. Only some LIN products support Master mode, see appropriate instrument handbook.

Note A fault, e.g. cable is disconnected or device loses power, will cause a lose of communications between the Master and Slave devices. Once the fault is fixed, Serial link communications can take up to 30 secs to be re-established. It can take up to 60 secs to re-establish Modbus TCP communications

By using the techniques of LIN function block caching, the Modbus Gateway facility can access data in other nodes distributed on the LIN, as well as LIN function blocks in the local LIN Database.

Note Some instruments support more than a single Modbus Gateway facility.

- Modbus - Serial

  The Serial communication interface type supports a single Master device communicating to any number of the Slave devices.

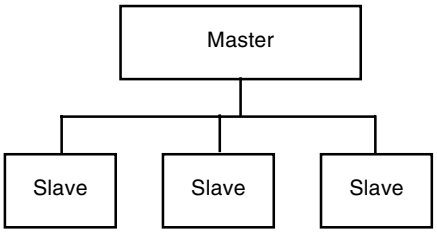


Figure 1.1a Serial communication interface type

- Modbus - TCP

  The Modbus TCP communication interface type supports multiple Master devices communicating to any number of the Slave devices.

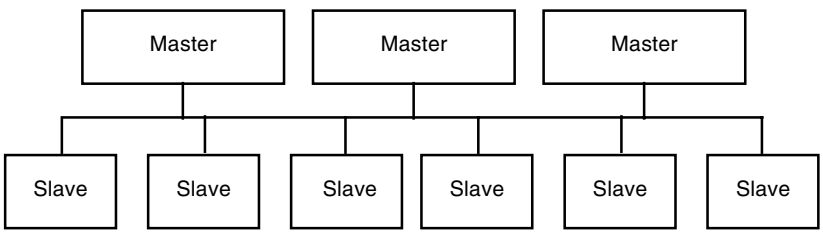


Figure 1.1b TCP/IP communication interface type

1.1.1 Main features

- The mapping between registers and function blocks is bidirectional.
- Multiple Modbus Gateway facility support. For devices that support more than one Modbus Gateway facility, mulitple GW_CON blocks will exist. Each GW_CON block defines a single GWF configuration that is to be run. An appropriate number of GW_TBL blocks may also be used if the features it contains are required to access Modbus diagnostic data.

  Note Any instrument that supports a single Modbus Gateway facility only does not support the use of the GW_CON block.

- The mapping between the LIN Database and the Modbus address space is entirely user-configurable for both digitals and registers.
- Digitals may be mapped as single bits, 8-bit bytes or 16-bit words.
- Analogue values map to single 16-bit registers with definable decimal point (Floating-point numbers as well as Integers.)
- 32-bit values (floating point or long integer, date and time) can be mapped to a pair of registers in some LIN products, see *Modbus Tools Online Help* (Part no. HA028988).

IMPORTANT 32-bit fields of storage must NOT run contiguously between different Tables.

- Modbus Tools software, see *Modbus Tools Online Help* (Part no. HA028988). The recommended package for configuring the Modbus interface and Tables, accessed via LINtools or the Start > Programs > ... > LINtools Advanced command on a PC.

  Note '...' indicates the installation path for the software.

- Modbus Configuration is supported via the *Modbus Tools* software. To configure the Modbus Gateway facility, simply fill in tables using prompts and menus to simplify the task. The validity of the entries are constantly checked during configuration time to minimise errors.
- Transparent Modbus Access (TMA or TalkThru), allows a PC, running a suitable client (typically iTools), to access a Slave device configuration, while connected to an instrument configured to operate in Master mode, see Transparent Modbus Access (TMA or TalkThru) section.
- Diagnostic and status registers allow the LIN Database to control the Modbus interface.
- The Modbus Gateway facility supports the Modbus RTU (8-bit) transmission mode.

  Note Modbus ASCII (7-bit) mode is not supported.

- The Modbus Gateway facility can be configured to support Comité de Normalisation des Moyens de production (CNOMO), i.e. it will respond to CNOMO registers with CNOMO data.

1.1.2 Functional description

The Modbus Gateway facility functions by keeping a copy of associated LIN Database parameters (cached block fields) in Modbus tables, that can be individually configured for either digital or register data. This copy of cached function block fields are updated from the LIN Database by a scanner task running in the Modbus Gateway facility. The scanner task identifies changes to specified block fields and updates the value held in the Modbus tables, which can then be read by the Master device at the next polling sequence request. This copy of cached function block fields in the Modbus Gateway does not reduce the space available for the continuous LIN Database.

Note The Modbus tables and communications port configurations can be configured using the Modbus Tools, see Modbus Tools Online Help (Part no. HA028988).

1.1.3 Function codes

The Modbus function codes, see Table 1.1.3, supported by the Modbus Gateway facility, together with their maximum scan counts, i.e. the maximum number of registers or bits that can be read or written in a single Modbus transmission of this type. For full details on Modbus messages and functions, see *Modbus Application Protocol Specification* (www.modbus-ida.org/specs.php).

Note *The Modbus Gateway facility makes no distinction between inputs and outputs. Thus any register or bit assigned in the Modbus Gateway facility can be accessed as both an input or an output as required. This follows the JBUS implementation of Modbus.*

Code	Function
1	Read digital output status
2	Read digital input status
3	Read output registers
4	Read input registers
5	Write single digital output
6	Write single output register
7	Fast read of single byte (*not configurable in Modbus Master*)
8	Diagnostics (*not configurable in Modbus Master*) (supports subcodes 0, 1, 2, 3, 4, A, C, D, E, F, 10, 11, 12 - see Table 1.4)
15	Write multiple digital outputs
16	Write multiple output registers
103	Dedicated TalkThru read output registers
106	Dedicated TalkThru write single output register

Note *Function codes 103 and 106 are manufacturer specific function codes.*

Table 1.1.3 Function codes

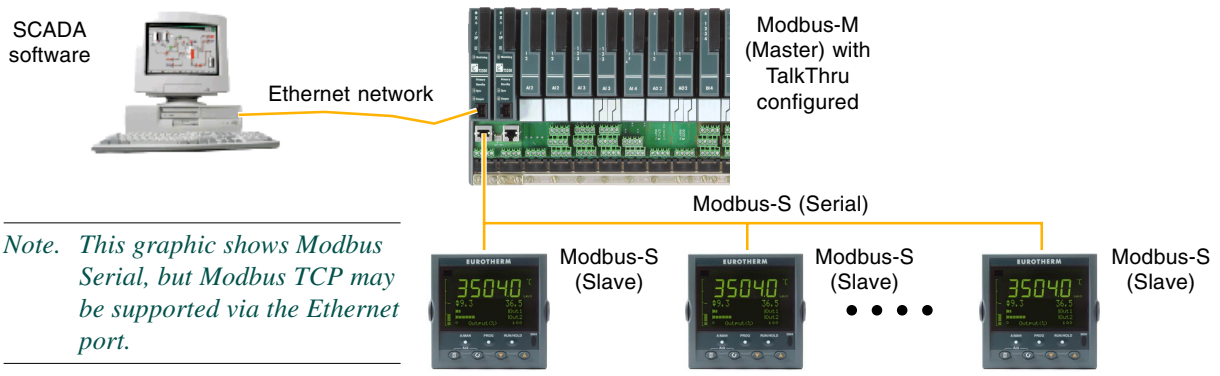
1.1.4 Transparent Modbus Access (TMA or TalkThru)

Transparent Modbus Access (TMA), or TalkThru, is a facility provided to enable the iTools package, see *iTools User Manual* (Part no. HA026179) or other third party Supervisory Control And Data Acquisition (SCADA) software to inspect and edit specific I/O parameters.

Note *iTools is capable of editing, storing and 'cloning' complete instrument configurations, as well as setting up data logging, process monitoring and Ethernet and modem connections.*

The iTools package operates on the PC and communicates with instruments via a network. It permits the configuration and monitoring of instruments operating in Slave mode by 'talking through' an instrument operating in Master mode. Access may be slower than if actually connected directly because the Master device is interleaving the TalkThru transactions, see *iTools User Manual* (Part no. HA026179) access with its own.

Note *Specific function codes have been allocated to enable the TalkThru facility.*



Note. *This graphic shows Modbus Serial, but Modbus TCP may be supported via the Ethernet port.*

Figure 1.1.4 Transparent Modbus Access (TMA or TalkThru) Configuration - example

1.2 PRINCIPLES OF OPERATION

The LIN Database groups related data into blocks, such as blocks of data representing an input, an output, or a controller etc. The LIN configurators and display packages recognise the different types of function block, and handle them appropriately. By contrast, the Modbus registers and bits (Modbus Gateway facility) are lists of data points in a device operating in either Master or Slave mode. In general there is no pre-defined structuring of these points into blocks or loops, etc., and most implementations define the allocation of registers differently.

Note Some instruments support Comité de Normalisation des Moyens de production (CNOMO) registers. These registers, offset parameters 121 to 124, return product details only when an instrument is operating in Slave mode.

Any Modbus Gateway facility involves the mapping of data from the instrument's LIN Database to Modbus registers and digitals. The mapping between registers and function blocks is bidirectional, see Figure 1.2, and is up to the Master device to manage how it interacts with a particular register or point. The mapping between the Modbus register and/or bits and the LIN Database is configured by the user.

Gaps can be left in the Modbus data areas for future expansion. These gaps can be written to and read from if required, allowing a system of 'letterboxes' to be set up that can be exploited by some systems. Data in the gaps does not interact with the standard LIN Database.

The Modbus Gateway facility functions by keeping a copy of associated function block fields in Modbus tables, that can be individually configured for either digital or register data. The Modbus table copy of function block fields are updated from the LIN Database by a scanner task running in the Modbus Gateway facility. The scanner task collects and compares the Modbus values and the LIN Database values. If the LIN Database parameter value has changed, the value is transferred to the associated register in the Modbus tables, so it can be read by the Master device at the next poll register request. If the cached block value in the Modbus tables has been changed by the Master device the value is transmitted to the LIN Database. The Modbus register table does not reduce the space available for the continuous LIN Database.

Note To maximise communications efficiency, dynamic data should be grouped so that it is available in contiguous table entries for a multi-parameter read.

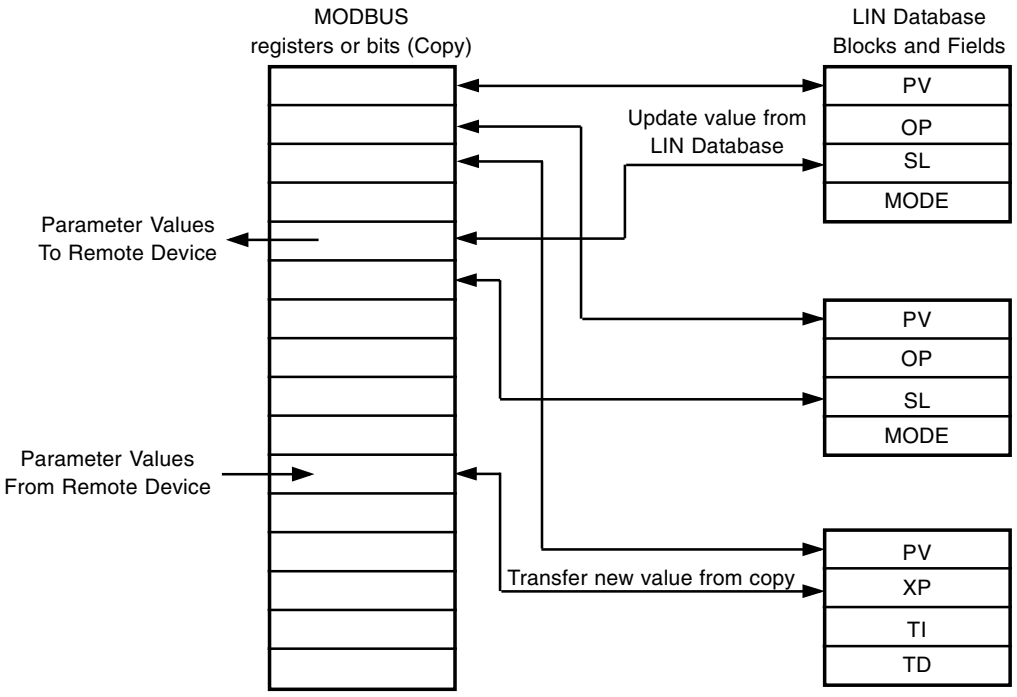


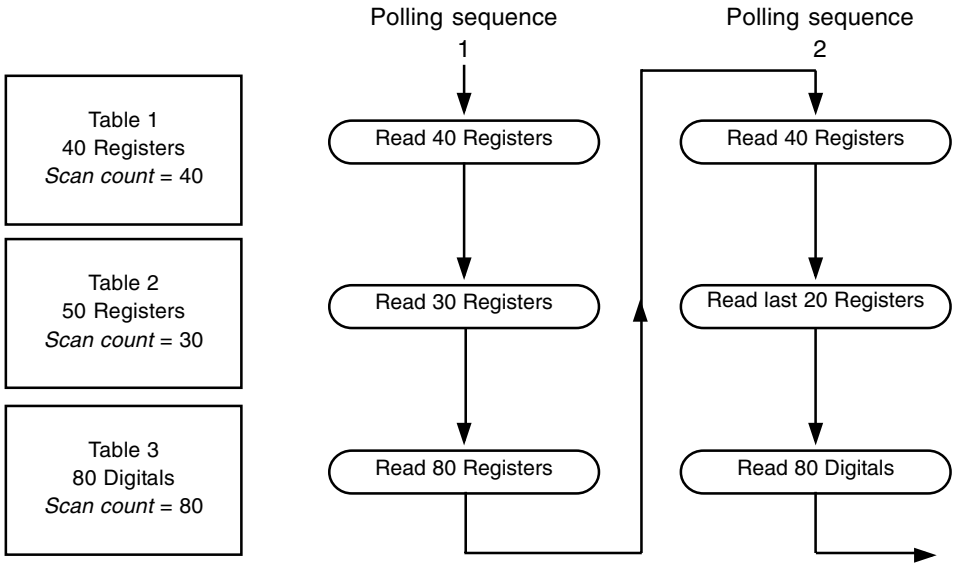
Figure 1.2 Modbus GateWay operation

1.2.1 Operating mode

MASTER MODE

LIN products configured to operate in Master mode can read and write values, to and from third party (Modbus communicating) devices operating in Slave mode controlled via a Polling sequence. Any LIN product configured to operate in Master mode that supports more than one Modbus Table, can communicate with the corresponding number of third party devices, up to 64, configured to operate in Slave mode.

Note This section is only applicable to devices that support Modbus Master mode operation, see appropriate instrument handbook.



Note. Table 2 shows the number of registers it contains (50) exceeds its maximum register count (30), so it takes two polling periods to be fully updated.

Figure 1.2.1 Polling period - example

Read operations

The Master cycles consecutively through the tables in the Modbus configuration at the configured *TickRate*, and polls each Slave allocated to these tables across the Modbus network. For each table, only one poll is made per sequence. If the Master cycle exceeds the configured *TickRate*, the *Status.TickSlip* bit of the GW_CON block is set TRUE for that table. The time to do a complete cycle of all the tables is called the Polling period. Thus, if a table is longer than the maximum count specified in the configuration, i.e. *Count* exceeds *Scan count*, it will take two or more Polling sequences to update all the data in that table. Clearly, if a table has to be read in several parts the overall Polling sequence of the Master cycle will be reduced.

Write operations

If the Scan task has detected that a value in one of the Slaves needs to be updated, it requests the Polling sequence to write the new value across the Modbus network. The Polling sequence is allowed to insert a maximum of *one* such write operation between consecutive read operations.

Note The example above shows that up to three writes could be made per Polling sequence.

SLAVE MODE

LIN products configured to operate in Slave mode, only allow the values in the Registers to be read by a Master device.

1.2.2 Refresh rates and timing information

The actual performance achieved by the Modbus Gateway Facility depends on many factors that exceed the scope or control of this manual, e.g. configured table scan rate, size and number of tables, quantity of data to transfer, and loading of the instrument processor. However, the actual performance achieved is shown in the *ScanPer* and *Period* fields of the GW_CON block or the *Diag11*, and *Diag12* fields in the MDBDIAG block, depending on the instrument type.

The total time taken for a change in the LIN Database of a Slave device to propagate via the communications link to the LIN Database of the Master device is the sum of the following,

$$\text{maximum propagation delay} = ss + ro + sm$$

where, *ss* = Table scan period between Modbus table and LIN Database in the Slave device, as shown in the *Period* field of the GW_CON Block in the Slave device

ro = Table read operations per master, generally once per *TickRate*. However, if the table read operation exceeds the configured *TickRate* it reverts to once per the *ScanPer,* as shown in the GW_CON block. This *ScanPer* value must be multiplied by the number of Polling sequences used to completely update the table.

sm = Table scan period between Modbus table and LIN Database in the Master device, as shown in the *Period* field of the GW_CON block in the Master device

1.2.3 Memory use and requirements

An area of memory is allocated to map the LIN Database parameters to the Modbus address space. This memory is allocated to tables, each table representing a series of consecutive registers or bits in the Modbus address space. The table contains an image of the data in the Modbus address space, and a Descriptor for each register, bit, or set of bits mapped onto that address space.

Note Each LIN product has specific configuration limits and memory requirements, see appropriate instrument handbook.

Requirement	Function
18 bytes per table	Overhead
2 bytes per register	Register Image Data
1-bit per digital	Digital Image Data. Automatically rounded up, see Digital Image Data section
6-bytes/entry register	Register Descriptor.
8-bytes/entry digital	Digital Descriptor.

Note Register and Digital Descriptor values are always applicable.

Table 1.2.3 Modbus address space requirements

REGISTER IMAGE DATA

The storage requirement of register image data is calculated using the Overhead, Image Data and Descriptor values.

Example:

- A register table with 40 register + 40 descriptors occupies:

18[overhead] + (40 x 2-bytes)[data] + (40 x 6-bytes)[descriptors] = 338 bytes.

DIGITAL IMAGE DATA

The storage requirement of digital image data is calculated by converting the total number of bits in the table to 8-bit bytes, then rounding this number of bytes up to the nearest **2-byte boundary**, i.e. the nearest even number. This means that a total bitcount from 1 to 16 will need 2-bytes of storage space, from 17 to 32 bits will need 4-bytes, from 33 to 48 bits will need 6-bytes, and so on.

The following formula can be used to provide an approximate value, assuming truncation and integer arithmetic:

2 x INT((*bitcount* + 15)/16) bytes.

The requirements for a digital table depend on how the data is mapped between the Modbus and the LIN Database. The examples below show the two extremes for mapping 64-bits to the LIN Database.

Examples:

- A digital table mapped onto the LIN Database in 16-bit units, needing only 4 descriptors occupies:

18[overhead] + 8[data] + (4 x 8)[descriptors] = 58 bytes.

- Each bit is separately mapped to a different point in the LIN Database in 16-bit units, needing a total of 64 descriptors occupies:

18[overhead] + 8[data] + (64 x 8)[descriptors] = 538 bytes.

1.2.4 Data conversion

The conversion of data between standard Modbus format and the LIN Database format is described here.

DATA CONVERSION OF DIGITALS

Modbus digital signals can be mapped onto LIN Database bitfields, booleans and alarms. The following rules apply to mapping these types into the Modbus address space.

- Bitfields can be mapped individually or as a complete set of 8- or 16-bits onto the Modbus address space.
- Booleans are mapped onto a single bit in the Modbus address space.
- Alarms are mapped onto a single bit in the Modbus address space. A value of '1' for this bit corresponds to the 'In alarm' status.

DATA CONVERSION OF REGISTERS

All data types can be mapped onto single registers in the Modbus address space. However, special care should be taken when mapping LIN Database values that require more than 16-bits, in particular 32-bit integers and floating point numbers.

Note Modbus Tables will become corrupt if registers containing 32-bit or 32-bit Swapped data type run contiguously between different Tables.

- Values requiring up to 16-bits of storage.

  LIN Database values that require up to 16-bits of storage (one or two bytes) are mapped directly onto a single register. This includes 8- and 16-bit integers, booleans, alarms and bitfields.

  Long signed 32-bit integers:

  When these values are transferred from the LIN Database to a Modbus register they are truncated, and only the low order 16-bits are written. When the register is being transferred from the Modbus to the LIN Database, the value is sign-extended into the high-order 16-bits.

  Long unsigned 32-bit integers:

  When these values are transferred from the LIN Database to a single Modbus register they are truncated, and only the low-order 16-bits are written. When the register is being transferred from the Modbus to the LIN Database, the high-order 16 bits are assumed to be zero.

  Floating-point numbers:

  When these values are transferred from the LIN Database to a Modbus register they are scaled according to the decimal point you specify, converted to an integer with rounding, limited to the range –65536 to +65535, and then truncated to 16 bits. This allows applications to work either with signed numbers (–32768 to +32767) or with unsigned numbers (0 to +65535).

  When the register is being transferred from the Modbus register to the LIN Database, it is treated as a signed number in the range –32768 to +32767, scaled according to the decimal point specified and then written to the LIN Database.

  CNOMO registers:

  The CNOMO registers apply to specific read-only Product data at pre-defined offsets. If an instrument is capable of a CNOMO response, the registers in the CNOMO range are already defined.

  Note The Modbus Tools software will not allow a LIN block parameter to be assigned to CNOMO registers.

1.2.4 DATA CONVERSION (Cont.)

- Values requiring up to 32-bits of storage.

  32-bit fields representing values where precision must be preserved may be connected to a pair of Modbus registers. The two parts are stored in standard PC format in two consecutive registers, of which the first must be at an even address. This method of linking is enabled by entering D (double precision) or S (swapped) in the DP field of the first register. The scanner task ensures data coherency.

  Note D, Double precision, is the least significant 16-bit word in the lowest offset register. S, Swapped, is the most significant 16-bit word in the lowest offset register.

  The instrument supports mapping of pairs of analogue registers to 32-bit IEEE format Word Swapped with low word first. If enabled then the 32-Bit - Swapped option is included on the Analogue register Format field in the Modbus Tools, see *Modbus Tools Online Help* (Part no. HA028988).

IMPORTANT 32-bit fields of storage must NOT run contiguously between different Tables.

  32-bit totals:

  Two-register mapping of long integers is used for the Total and Target fields of the TOTAL block and TOT_CONN block.

  Note This is only applicable to instruments that support the TOTAL block and TOT_CONN block.

  Date and Time:

  The Date and Time values can be transferred within the constraints of either POSIX or ISO8601 formats.

  POSIX format. This format maps both values into a single 32-bit number. The values are converted into the total number of seconds elapsed since midnight on January 1st 1970. When using this format, the time value is specified in the 'Field' column of the Modbus Tools. The rules for deriving these values are as follows:

  - If the referenced field is in the Configuration (header) block the Modbus table will directly map to the instrument's Real-Time Clock completely bypassing the LIN Database.

    Note All LIN Configuration (header) blocks have a TIME field, although not all have a DATE field.

  - For all function blocks, typically the DATE field is found immediately preceding the TIME field. However, if not found in the field immediately preceeding it, the field following it is interrogated. In the event that the DATE field is not found in either preceeding or following field, it will read as zero and ignore all writes.

  This format will support Modbus mapping from

  - instrument's own header block, including a T100 header block which does not have a date field
  - any cached header blocks, excluding a cached T100 header block
  - Date and Time values in BAT_CTRL block and SPP_CTRL block

    Note It also correctly rejects those TIME fields used for other purposes in SPP_CTRL, SPP_RAMP, and various DCM blocks as 'not time-of-day'. It does NOT support the TIMEDATE block.

  ISO8601 format. The ISO8601 format maps DATE and TIME fields separately.

  A DATE value is represented as a decimal number in the format YYYYMMDD. e.g. 14th November 2005 is represented as a decimal number, 20051114, and converted to a 32-bit hex value, 0x0131F4AA.

  A TIME value is represented as a decimal number in the format HHMMSS. e.g. 14:02:35 (2 minutes and 35 seconds past 2pm) is represented as the decimal number, 140235, and converted to a 32-bit hex value, 0x000223CB.

1.3 USING THE DIAGNOSTIC TABLE

The Diagnostic table is a special set of 16 offset parameters and an additional offset parameter for each Table, up to the maximum number of tables supported by that specific instrument. It allows the user to control the Modbus operation, or present diagnostic information to the LIN Database.

Note *Each entry in the Diagnostic Table is clearly illustrated on the Diagnostic page of the Modbus Tools software.*

Offset	Function
0	(*Unused*)
1	32-bit configuration error. Bit set if respective table has a invalid 32-bit ('D' or 'S') configuration. As there can be more tables than the 16-bits in this register, one bit covers multiple tables (e.g. with 64 tables, each bit covers 4 tables, bit 0 covers tables 1 - 4, etc.)
2	Diagnostic register, bits currently allocated: Bit 5 - Slave in listen-only mode
3	Query data as transmitted by function code 8 sub code 0
4	Input delimiter as transmitted by function code 8 sub code 3
5	Count of slave responses that do not match the request (master mode only)
6	CRC errors count. Count of received messages containing CRC errors
7	Count of exception responses received from Modbus Master (master mode) sent by Modbus Slave (slave mode)
8	Count of received good messages
9	Count of messages for which the slave did not respond (master mode only)
10	Count of received bad characters
11	Master polling task: cycle period
12	Scanner task: time to check all tables
13	Scanner task: time used last time scheduled
14	Scanner task: time used for last delay
15	(*Unused*)
16 to *n*	Entry for each corresponding table, where *n* equals the maximum number of tables supported by the instrument

These periods are instrument specific, see appropriate instrument handbook

Table 1.3.1 Diagnostic table

The offset parameters of a diagnostic table are in two distinct sets. The first sixteen, see internal diagnostic registers section, at offset 0 to 15 contain general information on the independent operating mode of the Modbus instrument. Each of the next offset, 16 to *n*, see Modbus table status and control registers section, contain status and control bits that allow the LIN Database to interact with the Modbus tables, e.g. the T2550 instrument has a total of 80 diagnostic offset parameters, 16 Internal diagnostic registers and 64 separate offset parameters for each Modbus Table.

Note *Instruments that support only a single Modbus Gateway facility can use the MDBDIAG block, see LIN Block Reference Manual (Part no. HA082375U003) to access Modbus diagnostic data. This will release one table from the configuration and disable the corresponding Diagnostic table, thereby avoiding bit value conflicts.*

1.3.1 Internal diagnostic registers

The first set of offset parameters (with default offset 0 to 15) are for internal diagnostic use, see Table 1.3.1, and are read-only to the user. They present general information on the operation of the Modbus, and their functions are independent of whether the instrument is operating in Master mode or Slave mode.

1.3.2 Modbus table status and control registers

The second set of offset parameters (with default offset 16 to *n*, where *n* equals the last configured Modbus table) allow individual tables in the configuration to be monitored and controlled. Each offset in the diagnostic table is automatically allocated to an entire table in the configuration. Specifically, offset 16 is assigned to Table 1, offset 17 is assigned to Table 2, and so on.

The functions of this second set of registers depends on whether the LIN product is operating in Master mode or Slave mode.

1.3.3 Diagnostic table registers

The Diagnostic register includes bits that allow control by a LIN Database Sequence of read/write operations when required by the application, see Figure 1.3.3.

Note Some instruments do not support all bits shown below.

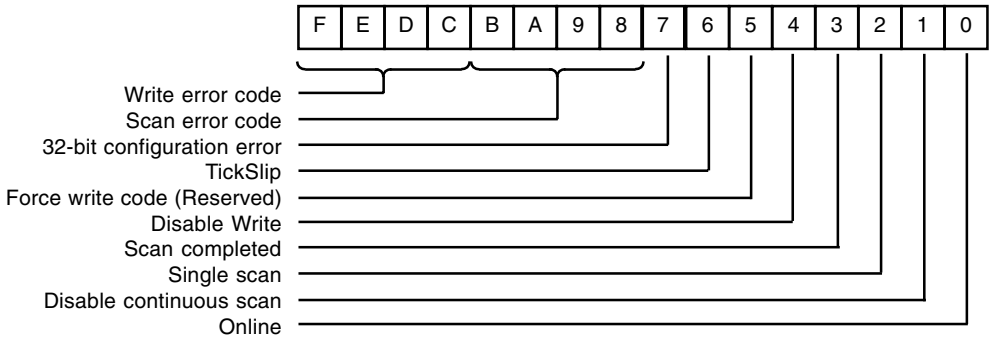


Figure 1.3.3 Diagnostic table registers

Diagnostic Table Register	Bit Code	Description
Write error code	wrErr	Normally zero. Otherwise it contains the error code associated with the last write to this table.
Scan error code	scanErr	Normally zero. Otherwise it contains an error code associated with the reading of this table.
32-bit configuration error*	Cfg32Err	Sets if respective table has a invalid 32-bit ('D' or 'S') configuration.
TickSlip	TickSlip	Sets if respective table is unable to scan table at the requested Tick rate.
Force write code (Reserved)	forceWr	Internal bit used to force a write to the respective table.
Disable write*	DisWr	If set to 1, it will stop the Master writing to the Slave across the MODBUS network. When reset to 0, a write is forced to ALL the values in the table. The slave will return error code 8, see Table 1.5, Exception responses from a Modbus Slave.
Scan completed	complete	If set to 1, the Master has completed a scan of the Slave. When operating in single scan mode, it indicates the scan is finished and the data is available for use, and is reset when the single scan bit is set.
Single scan	single	Set in conjunction with the *disable continuous scan* bit allowing a LIN Database sequence to start a single scan.
Disable continuous scan	disScan	If set to 1, the Master will stop polling the Slave across the Modbus link. The *single scan* and *scan completed* bits are used together when an instrument can be polled only under specific circumstances. A simple LIN Sequence can be implemented to ensure that these bits are used correctly. The *Disable continuous scan* bit **must** also be set. The suggested sequence for these operations is: **1** Reset the *Single Scan* bit **2** Wait for *Scan completed* reset **3** Set the *Single Scan* bit **4** Wait for Scan completed set **5** The data is now valid **6** Loop back to step 1
Online*	Online	If the instrument is operating in Modbus Slave mode this bit is set to 1 when the table has been written to or read from within the period defined in *Time out*. If the instrument is operating in Modbus Master mode the instrument operating in Slave mode has failed to respond within the period defined in *Time out* and all retries have failed.

*Note. * indicates the code applies to instruments operating in Modbus Master or Modbus Slave mode.*

Table 1.3.3 Modbus diagnostic register codes

1.4 DIAGNOSTIC FUNCTION CODES

Table 1.4 summarises how the common Modbus diagnostic function codes have been supported by the LIN products configured to operate as a Modbus Slave. The diagnostics are accessed via Modbus function code 8.

Diagnostic Sub-Code	Data Sent	Description
0000	xxxx	Echoes the data sent
0001	0000	Restarts
	FF00	Resets the diagnostic counters, and re-enables responses if the slave had been placed in Listen-only mode by sub-code 4.
0002	xxxx	Returns the diagnostic register. (In the current versions, the returned data is always zero.)
0003	ABxx	Changes ASCII delimiter. (This echoes the data sent.)
0004	0000	Forces Listen-only mode. There is NO response to this function.
000A	0000	Resets all counters.
000B		(*Not supported*)
000C	0000	Returns the number of CRC errors detected in messages addressed to this slave.
000D	0000	Returns the number of error messages returned by this slave.
000E	0000	Returns the number of correct messages addressed to this slave.
000F	0000	Returns a count of the number of times the slave has not responded to a valid message (e.g. due to an unsupported function, or a buffering problem in the slave).
0010	0000	Always returns 0.
0011	0000	Always returns 0.
0012	0000	Returns the count of character errors received at the slave, i.e. (overrun + parity + framing) errors.
0013		(*Not supported*)
0014		(*Not supported*)

Table 1.4 Modbus diagnostic function codes

1.5 MODBUS EXCEPTION RESPONSES

Table 1.5 lists all the error codes that may be returned in an exception response from a LIN product configured to operate in Slave mode.

Code	Data	Description
01*	Illegal function	The function is illegal, or not supported within the instrument
02*	Illegal data address	The address referenced does not exist in the slave device
03*	Illegal data value	The value in the data field is invalid
04	Failure in associated device	
05	Acknowledge	
06	Busy, rejected message	
07	NAK-negative acknowledgement	
08*	Write error	The data has been write-protected via a bit in the appropriate table diagnostic register.
09	Zone overlap	
0A	Path Unavailable	Gateway Path Unavailable. Associated Master not running, TalkThru failed.
0B	Slave absent	Gateway Target Device failed to respond. TalkThru device offline, TalkThru failed.
0C	CRC error	
0D	Transmission blocked/Timeout	Scan Error. The data has not been updated, within the specified period.

Note. The current Modbus standard definition for Exception response code 08 is 'Memory parity error'.

Table 1.5 Exception responses from a Slave device

1.6 NOTES ON MODBUS IMPLEMENTATION

Although based on the original Modbus specification, different manufacturers' implementations vary slightly in the correspondence between the actual register or bit addresses in a PLC, for example, and the Modbus/JBUS address, i.e. the 'protocol address'. It is this protocol address that is to be configured in the Modbus Gateway facility implementation.

Note For details about the operation of third party products, refer to the appropriate manufacturers instrument documentation.

1.6.1 Modbus (AEG-MODICON) implementation

Read-only (input) and read/write (output) registers and bits are assigned to separate tables, each with its own address-offset relative to the Modbus protocol address, see Table 1.6.1.

It is the Modbus function code that determines the value of the offset required, and therefore whether a given Modbus protocol address is directed at an input or output, in a bit or register table.

Data Type	Modbus Function Codes Read	Write	PLC Address	Protocol Address
Output bits	01	05, 15	00001 + X	X
Input bits	02	N/A	10001 + X	X
Output registers	03	06, 16 (103, 106)	40001 + X	X
Input registers	04	N/A	30001 + X	X

Note. Read/write (output) register function codes 103 and 106 support the TalkThru facility.

Table 1.6.1 PLC address offsets for different data types

1.6.2 JBUS implementation

The JBUS protocol is identical in all respects but one to the Modbus protocol. The one difference concerns the parameter or register address. Both use a numeric index but the JBUS index starts at '0' while the Modbus index starts at '1'.

1.6.3 Other product implementations

Other manufacturers' 'Modbus Gateway' implementations conform to the MODICON principle of separate tables for different types of data exchange, but the correspondence between PLC base address and Modbus protocol address is user-configurable.

Intentionally left blank

CHAPTER 2 MODBUS DCM

This section describes the implementation of the Modbus Devolved Control Module (DCM) as a part of a LIN instrument, in the following subsections:

Note Some instruments act as a Modbus Master when running Modbus DCM.

- Introduction (Section 2.1)
- Instrument Configuration (Section 2.2)
- The .UYM File (Section 2.3)

2.1 INTRODUCTION

A Devolved Control Module (DCM) must be configured for each item to be accessed via the Modbus link. In addition, an Instrument function block is available for each model of I/O unit produced by the manufacturer. This contains various instrument and Modbus parameters along with instrument and I/O failure and status indications.

When all function blocks have been configured correctly, and are resident in the LIN Database along with all the appropriate files and any relevant Universal Map for Modbus (.uym) files, see The .UYM File section, the instrument will start communicating with the I/O unit as soon as it is initialised, without the need to set up mapping tables as is required by the Modbus Gateway version.

For 'third party' instruments, a .uym file must be created for each DCM.

Note All the DCMs function blocks are described in DCM section of the LIN Blocks Reference Manual (Part no. HA082375U003).

2.2 INSTRUMENT CONFIGURATION

Instruments use files to retain a set of instrument and communication parameters specific to each LIN product.

- Visualisation Instruments

  These are instruments, e.g. T800, T801, T802, and T820, that are used to show subsystem data via a series of screen displays. The instrument configuration must be edited using either the on-screen menus or the Instrument Properties dialog, see *Instrument Properties Online Help*.

- Process Instruments

  These are instruments, e.g. T940X, and T2550, that are used to control the process values of a subsystem. The instrument configuration must be edited using the Instrument Properties dialog.

> *Note* *After changes to the Instrument Properties have been downloaded, some instruments must be restarted before the changes can be implemented.*

2.2.1 Instrument Properties

The Instrument Properties are shown on the Instrument Properties dialog. This should be used, in conjunction with the other software tools available to ensure the appropriate Instrument parameters are correctly defined. The Instrument Properties are divided into two groups and displayed as property pages from within an application, e.g. LINtools, or by selecting the Properties command in Windows Explorer.

- Instrument Options

  These parameters define the instrument configuration.

- Network Settings

  These parameters define the position of an instrument on a Local Instrument Network, LIN.

The Instrument Properties dialog can,

- read/write changes into the instrument filesystem
- organise and present possible configuration options to the user
- communicate with online instruments in order to both read their current options settings and download modified instrument parameters
- instruct an instrument (via LIN messages) to reload any Instrument filesystem settings that do not require the power to first be isolated then re-applied
- present the user with only the configuration options applicable to the target instrument

> *Note* *The Instrument Properties dialog can be launched from both LINtools and Windows Explorer. Any changes to the Instrument Properties dialog will be automatically reloaded and subsequently, update the instrument filesystem.*

2.2.2 Mapping Files

As part of the LIN Database, DCM function blocks are used to specify a particular point in the subsystem. A Universal Map for Modbus (.uym) file is required for every DCM which is to communicate with a Slave device, in order that the instrument may know the address at which a particular point (channel value, alarm threshold value etc.) is to be found.

2.3 THE .UYM FILE

A .uym file is required for every DCM which is to communicate with a Slave device, in order that the instrument may know the address at which a particular point (channel value, alarm threshold value etc.) is to be found. This information has be obtained from the Slave device documentation.

Notes

1. *An unlimited number of function blocks may use the same .uym file, provided only that their register usage (i.e. the range of addresses) in the target instrument is identical.*
2. *.UYM files can also be used to override information in Standard DCMs.*

The .uym file, see DCM section of the *LIN Blocks Reference Manual* (Part no. HA082375U003), can be edited using an appropriate text editor, i.e. 'notepad.exe', and loaded into the LIN Database. The format is as follows:

```
Field,Register,Type,Function codes,Ranges (for normalised types)
```

Note Underlined items do not need to be included if the defaults are acceptable.

Field — The name of the function block being mapped.

Register — The required Modbus register of the point being accessed.

Register can be a simple decimal number or it can be of the form:

```
Constant1[Constant2*(Field name ± Constant3)]
```

Where:

Constants 1, 2 and 3 are simple decimal numbers,

Field name is any name in the function block that has a 16-bit integer value. A sample expression might be:

```
200 + 10[(Slot_No + Chan_No)-1]
```

In which Constants 1, 2 and 3 are 200, 10 and 1 respectively, and the field name is

'Slot_No + Chan_No'.

Type — Number type (data format). This field needs to be entered only if the default (Unsigned Integer (UINT)) is not the correct type, see Table 2.3 for number type (data format) entries.

Function codes — Modbus function codes. This needs to be entered only if the default (3, 4 - read registers) is not acceptable, see Table 1.1.3 for a list of supported Function codes. Setting the value to zero disables the item, i.e. it will not communicate. This feature is provided to allow the disabling of communications for individual fields in standard DCM function blocks.

Ranges for normalised types — This is the pair of values (as <min>:<max>) of which the 16-bit value obtained from the remote node is scaled to convert it to a floating point value in the LIN Database, e.g. 0:100. For a NORM type <min> is the floating point value equivalent of 0 in the register and <max> is the equivalent of 32767. For a UNORM type <min> is the floating point value equivalent of 0 in the register and <max> is the equivalent of 65535.

2.3 THE .UYM FILE (Cont.)

Table 2.3 shows the supported Number types (data format).

Number Types	Definition
BOOL	Value 0/1 in least significant bit
UINT	Unsigned 16-bit integer
INT	Signed 16-bit integer
USINT	Unsigned 8-bit integer
SINT	Signed 8-bit integer
UDINT	Unsigned 32-bit integer
UDINT_X	Unsigned 32-bit integer (*See Note*)
DINT	Signed 32-bit integer
DINT_X	Signed 32-bit integer (S*ee Note*)
TIME	Signed 32-bit duration in milliseconds
STIME_ds	16-bit duration in deciseconds (0.1s)
STIME_dm	16-bit duration in deciminutes (0.1m)
STIME_dh	16-bit duration in decihours (0.1h)
REAL	32-bit IEEE floating point value in 2 registers
REAL_X	32-bit IEEE floating point value in 2 registers (**See NOTE*)
SREAL_p1	16-bit number in units of 0.1
SREAL_p2	16-bit number in units of 0.01
SREAL_p3	16-bit number in units of 0.001
SREAL_p4	16-bit number in units of 0.0001
SUREAL_p1	16-bit Unsigned number in units of 0.1
SUREAL_p2	16-bit Unsigned number in units of 0.01
SUREAL_p3	16-bit Unsigned number in units of 0.001
SUREAL_p4	16-bit Unsigned number in units of 0.0001

Note *These Number Types (data formats) have the pair of 16-bit words in the reverse order compared to the corresponding non-'_X' formats. They are provided specifically for communicating with other LIN instruments via the Modbus Slave Gateway.*

Table 2.3 Supported Number types

Example:

The function of this example is to read an analogue input value from a channel of an instrument.

Note *To clarify the example, attempting to read an analogue input value from a channel 17 of a recorder.*

The Communications parameters, have been set up in the Configuration: Comms menus, as follows (to match the instrument settings):

Protocol:	MODBUS
Baud Rate:	9600
Parity:	Even
Data bits:	8 (fixed for MODBUS protocol)
Stop bits:	1
H/W handshake:	Off
Address:	4

From the instrument documentation, the analogue input channels are accessed using Function Code 03 and are addressed contiguously, starting with channel 1 at decimal address 0. Thus to read its input value, Function Code 03 and address 16 are required.

The sample .uym file should contain the following:

```
MV,16,UINT,"03"
```

2.3 THE .UYM FILE (Cont.)

2.3.1 Scaling

The instrument documentation also states that the value (PV) is returned as a 16-bit hex number in the range 0000 (Channel Low range value) to FFFF (Channel High range value), and the calculation:

$$Scaled\ value = \left((High\ range - low\ range) \times \frac{PV}{FFFF} \right) + low\ range$$

has to be carried out to find the actual scaled value. The instrument Channel Configuration must be accessed to determine the High and Low range values.

Example:

High range = 90% for 4V input signal

Low range = 10% for 1V input signal

Current PV = 2.5V (7FFF)

The scaled value is

{(90 - 10)% x 7FFF/FFFF} + 10% = 50%

2.3.2 Commenting

Note To clarify the example, attempting to read an analogue input value from a channel 17 of a recorder.

A comment can be attached to the end of one or more lines in the form:

```
, , "Comment"
```

The maximum number of characters for the .uym file is 60 characters, including delimiters. The comment text string can contain a maximum of (60 minus rest of line) characters.

The sample .uym file could become:

```
MV,16,UINT,"03",,"Recorder 4, channel 17"
```

Intentionally left blank

CHAPTER 3 PROFIBUS

This section describes the implementation of the Profibus facility as a part of a LIN product, in the following subsections:

- Introduction (Section 3.1)
- Instrument Configuration (Section 3.2)
- Devolved Control Modules (DCMs) (Section 3.3)
- The .UYP file (Section 3.4)
- Installation (Section 3.5)
- Troubleshooting (Section 3.6)
- Global commands (Section 3.7)
- Operation (Section 3.8)
- The .GSD file (Section 3.9)
- Redundant (Duplex) mode (Section 3.10)

3.1 INTRODUCTION

PROFIBUS DP is an industry standard open network used to interconnect instrumentation and control devices in, for example, a manufacturing or processing plant. It is often used to allow a central Programmable Logic Controller (PLC) or PC based control system to use external 'slave' devices for input/output (I/O) or specialised functions, thus reducing the processing load on the controlling unit so that its other functions can be carried out more efficiently, using less memory.

This implementation of the PROFIBUS network uses a high speed version of the EIA485 standard to permit transmission rates of up to 12Mbits/second between the host and multiple 'Stations' otherwise called 'nodes' either within a single section of network or, with EIA485 repeaters, in several separate sections of network. Acceptable node addresses are 3 to 126.

It is not within the scope of this document to describe the PROFIBUS standard in detail; this information can be found by reference to the Profibus web site:

'http://www.profibus.com'.

3.2 INSTRUMENT CONFIGURATION

Instruments use files to retain a set of instrument and communication parameters specific to each LIN product.

- Visualisation Instruments

  These are instruments, e.g. T800, T801, T802, and T820, that are used to show subsystem data via a series of screen displays. The instrument configuration must be edited using either the on-screen menus or the Instrument Properties dialog, see *Instrument Properties Online Help*.

- Process Instruments

  These are instruments, e.g. T940X, and T2550, that are used to control the process values of a subsystem. The instrument configuration must be edited using the Instrument Properties dialog.

> *Note* *After changes to the Instrument Properties have been downloaded, some instruments must be restarted before the changes can be implemented.*

3.2.1 Instrument Properties

The Instrument Properties are shown on the Instrument Properties dialog. This should be used, in conjunction with the other software tools available to ensure the appropriate Instrument parameters are correctly defined. The Instrument Properties are divided into two groups and displayed as property pages from within an application, e.g. LINtools, or by selecting the Properties command in Windows Explorer.

- Instrument Options

  These parameters define the instrument configuration.

- Network Settings

  These parameters define the position of an instrument on a Local Instrument Network, LIN.

The Instrument Properties dialog can,

- read/write changes into the instrument library file
- organise and present possible configuration options to the user
- communicate with online instruments in order to both read their current options settings and download modified instrument parameters
- instruct an instrument (via LIN messages) to reload any Instrument library file settings that does not require the power to first be isolated then re-applied
- present the user with only the configuration options applicable to the target instrument

> *Note* *The Instrument Properties dialog can be launched from both LINtools and Windows Explorer. Any changes to the Instrument Properties dialog will be automatically reloaded and subsequently, update the instrument filesystem.*

3.2.2 Mapping Files

As part of the LIN Database, DCM function blocks are used to specify a particular point in the subsystem. A Universal Map for Profibus (.uyp) file is required for every DCM which is to communicate with a Profibus slave device, in order that the instrument may know the address at which a particular point (channel value, alarm threshold value etc.) is to be found.

3.3 DEVOLVED CONTROL MODULES (DCMS)

A Devolved Control Module (DCM) must be configured for each item to be accessed via the Profibus link. In addition, an Instrument function block is available for each model of I/O unit produced by the manufacturer. This contains various instrument and Profibus parameters along with instrument and I/O failure and status indications.

When all function blocks have been configured correctly, and are resident in the LIN Database along with the all appropriate hidden library files and any relevant Universal Map for Profibus (.uyp) files, see The .UYP File section, then the instrument will start communicating with the I/O unit as soon as it is initialised.

For 'third party' instruments, a .uyp file must be created for each DCM.

Note *The maximum number of digital input modules supported by each LIN product will differ. All the DCMs function blocks are described in DCM section of the LIN Blocks Reference Manual (Part no. HA082375U003).*

3.4 THE .UYP FILE

3.4.1 Introduction

A .uyp file is required for each DCM function block communicating with a Slave device. It is used by the instrument to define the address of a particular point (channel value, alarm threshold value etc.). The point address is obtained from the documentation supplied with the slave device.

Note An unlimited number of function blocks may use the same .uyp file, provided only that their register usage (i.e. the range of addresses) in the target instrument is identical.

The .uyp text file is loaded into the machine's filesystem, see DCM section of the *LIN Blocks Reference Manual* (Part no. HA082375U003). The format is as follows:

```
Field,Address,Type,"Operations",Ranges (for normalised types)
```

Note Underlined items do not need to be included if the defaults are acceptable

Field	The name of the function block field being mapped.

Address	The required register of the point being accessed. This address can be a simple decimal number or it can be of the form:
	`Constant1 + Constant2*(Item name ± Constant3)`
	Where:
	Constants 1, 2 and 3 are a simple decimal numbers,
	Field name is any name in the function block which has a 16-bit integer value. A sample expression might be:
	`200 + 10[(Slot_No +Chan_No)-1]`
	In which Constants 1, 2 and 3 are 200, 10 and 1 respectively, and the field name is
	`'Slot_No + Chan_No'.`

Type	The number type. This needs to be entered only if the default (Unsigned Integer (UINT)) is not the correct type, see Table 2.3 for number type (data format) entries.
Operations	One or more of: VO (Void = no data transfer. Used to remove items from standard DCMs.) RC, WC, RA, WA, where R = read, W = write, C = cyclic, A = acyclic Default = RC (read cyclic)
Ranges (for normalised types)	This is the pair of values (as <min>:<max>)of which the 16-bit value obtained from the remote node is scaled to convert it to a floating point value in the LIN Database, e.g. 0:100. For a NORM type <min> is the floating point value equivalent of 0 in the register and <max> is the equivalent of 32767. For a UNORM type <min> is the floating point value equivalent of 0 in the register and <max> is the equivalent of 65767.

3.4 THE .UYP FILE (Cont.)

Example:

The following is an example of how .uyp files relate to the cyclic data for a fictitious slave called 'PLC 1'. The Module string for this slave is:

```
Module = "PLC 1" 0x00,0x50, 0x60, 0x71, 0x95
```

Endmodule.

The interpretation of the module identifiers (e.g. 0x71) is given in the table below. Note that the hex numbers are required in BCD for this interpretation. For example 0x71 is decoded as 0111 0001, and 0x95 as 1001 0101. Bit 0 is the least significant (i.e. right-most) bit.

Bit Number	Identifier Interpretation
Bit 7 (MSB)	C
Bit 6	U
Bit 5	O
Bit 4	I
Bit 3	L3
Bit 2	L2
Bit 1	L1
Bit 0 (LSB)	L0

Table 3.4.1a Module identifier interpretation

MODULE IDENTIFIER BIT DEFINITION

Bit 7	C	Set (1) =	Consistency over module
		Clear (0) =	Consistency over units
Bit 6	U	Set (1) =	L3 to L0 in words
		Clear (0) =	L3 to L0 in Bytes
Bit 5	O	Set (1) =	Module has data in write direction
		Clear (0) =	No data in write direction for the module
Bit 4	I	Set (1) =	Module has data in Read direction
		Clear (0) =	No data in Read direction for the module
Bit 3 to Bit 0	L3 to L0	Add one to this value to calculate the length of the cyclic data for the module, in words (Bit 6 set) or in Bytes (Bit 6 clear)	

Examples:

0x50 (0101 0000)	The module units setting = words. Data direction = Read. L3 to L0 = 0, so data occupies 1 word (2 Bytes) in the read cyclic area.
0x60 (0110 0000)	The module units setting = words. Data direction = Write. L3 to L0 = 0, so data occupies 1 word (= 2 Bytes) in the write cyclic area.
0x71 (0111 0001)	The module units setting = words. Data direction = Read/Write. L3 to L0 = 1, so data occupies 2 words (= 4 Bytes) in both read cyclic and write cyclic areas.
0x95 (1001 0101)	The module units setting = Bytes. Data direction = Read. L3 to L0 = 5, so data occupies 6 bytes in the read cyclic area
0x00 (0000 0000)	Transacts no data. Normally used to signify an empty module slot in a modular system, see 'Special Module Identifiers.

3.4 THE .UYP FILE (Cont.)

SPECIAL MODULE IDENTIFIERS

The format for special module identifiers is shown in Table 3.4.1b

Bit Number	Identifier Interpretation Byte 0	Byte 1	Byte 2	Byte M0 to Mn
Bit 7 (MSB)	O	Co	Ci	Manufacturer Specific Data
Bit 6	I	Uo	Ui	
Bit 5	0	Lo5	Li4	
Bit 4	0	Lo4	Li4	
Bit 3	M3	Lo3	Li3	
Bit 2	M2	Lo2	Li2	
Bit 1	M1	Lo1	Li1	
Bit 0 (LSB)	M0	Lo0	Li0	

Table 3.4.1b Special Module identifier interpretation

Byte 0 bit 7	O	Set (1) = Module has data in Write direction. Byte 1 is present. Clear (0) = No module data in Write direction. Byte 1 absent.
Byte 0 bit 6	I	Set (1) = Module has data in Read direction. Byte 2 is present. Clear (0) = No module data in Write direction. Byte 2 absent.
Byte 0 bits 3 to 0	M3 to M0	Gives the number of bytes of Manufacturer-specific data (max. 14).
Byte 1 bit 7	Co	Set (1) = Consistency over output module. Clear (0) = Consistency over output units.
Byte 1 bit 6	Uo	Set (1) = Lo5 to Lo0 in words. Clear (0) = Lo5 to Lo0 in Bytes.
Byte 1 bits 5 to 0	Lo5 to Lo0	Add one to this value to calculate the length of the cyclic write data for the module, in words (Byte 1 bit 6 set) or in Bytes (Byte 1 bit 6 clear).
Byte 2 bit 7	Ci	Set (1) = Consistency over input module. Clear (0) = Consistency over input units.
Byte 2 bit 6	Ui	Set (1) = Li5 to Li0 in words. Clear (0) = Li5 to Li0 in Bytes.
Byte 2 bits 5 to 0	Li5 to Li0	Add one to this value to calculate the length of the cyclic read data for the module, in words (Byte 2 bit 6 set) or in Bytes (Byte 2 bit 6 clear).
Bytes M0 to Mn		A number of bytes of manufacturer-specific data. The number of bytes is given by Byte 0 Bits 3 to 0 (M3 to M0).

Examples:

The following is the interpretation of the Module String

Module = “PLC 1” 0xC3, 0xCF, 0xC7, 0x21,0x22, 0x23

End Module

0xC3 (1100 0011)	Data direction = Read/Write. M3 to M0 = 3, so there are 3 bytes of manufacture-specific data.
0xCF (1100 1111)	Write cyclic data modules are consistent over their entire length. Units are words. Lo5 to Lo0 = 15 words, so data occupies 16 words (= 32 Bytes) in the write cyclic area.
0xC7 (1100 0111)	Read cyclic data modules are consistent over their entire length. Units are words. Lo5 to Lo0 = 7 words, so data occupies 8 words (= 16 Bytes) in the write cyclic area.
0x21 to 0x23	Manufacturer-specific data.

3.4 THE .UYP FILE (Cont.)

The following tables define the structure for the cyclic Write data and the cyclic Read data for our fictitious slave:

```
Module = "PLC 1" 0x00,0x50, 0x60, 0x71, 0x95
```

Endmodule.

Module	Size	Offset	Structure	Interpretation
0x60	2	0	Word	Output enable
0x71	4	2	Word	Sensor enable
		4	Word	LED on

Table 3.4.3c Cyclic Write data structure

Module	Size	Offset	Structure	Interpretation
0x00	0	0	N/A	Empty module
0x50	2	0	Word	Input status
0x71	4	2	Word	Sensor status
		4	Word	Sensor type
0x95	6	6	Byte	Input under range
		7	Real	Input value
		11	Byte	Input over range

Table 3.4.3d Cyclic Read data structure

Notes

1. *The meaning of the data is not defined in the .gsd file, it must be obtained from the slave's documentation.*
2. *The offset is the address of the first Byte of the item with respect to the start of the slave's data in the Profibus memory image. Read and write offsets are independent of one another.*

UYP FILES FOR THE SLAVE 'PLC 1'

The .uyp files for our fictitious Slave devices can now be written. The purpose of the .uyp file is to inform the LIN Database where the data for a generic DCM is to be found in the Profibus memory image. Each generic DCM contains the name of the .uyp file it must use to access its data.

Notes

1. *Each generic DCM can use only a single .uyp file.*
2. *Generic DCMs are specific to a particular data type.*
3. *A .uyp file can be used by only one DCM.*

3.4 THE .UYP FILE (Cont.)

This fictitious Slave device of our example requires 5 .uyp files: four for cyclic read, and 1 for cyclic write.

Example:

```
CYCLIC WRITE

Both modules can be included in one file, because the number type is the same (Word = Int)
for all the data. Thus the cyclic write uym can be written as follows:

File0.uyp

Op_En,0,INT,"WC"

Sens_En,2,Int,"WC"

LED_On,4,INT,"WC".


In the above,

'Op_En', 'Sens_En' and 'LED_On' are descriptive strings displayed in the DCM when the
database is running,

'0', '2' and '4' are the respective offsets of the data in the slave's memory image.

INT means 16-bit integer (two Bytes).

WC means Write Cyclic


CYCLIC READ


File1.uyp

Ip_Stat,0,INT,"RC"

Sen_Stat,2,INT,"RC"

Sen_Type,4,INT,"RC"

File2.uyp

Ip_low,6,USINT,"RC"

File3.uyp

InputDat,7,REAL,"RC"

File4.uyp

Ip_high,11,USINT,"RC"

In the above,

'Ip_Stat', 'Sen_Stat', 'Sen_Type', 'Ip_low', 'InputDat' and 'Ip_high' are all descriptive
strings displayed in the DCM when the database is running,

'0', '2', '4', '6',  '7' and '11' are the respective offsets of the data in the slave's
memory image.  Each item follows immediately after the preceding one.

INT means 16-bit (two Byte) integer, USINT means unsigned short integer (8-bits = 1 Byte))
and REAL means 32-bit (four Byte) representation ofthe input value.

RC means Read Cyclic.
```

3.4.2 Commenting

Note To clarify the example, attempting to read an analogue input value from a channel 17 of a recorder.

A comment can be attached to the end of one or more lines in the form:

```
, , "Comment"
```

The maximum number of characters for the .uyp line is 60 characters, including delimeters. The comment text string can contain a maximum of (60 minus rest of line) characters.

Example:

The sample .uyp file could be,

```
MV,16,UINT,"RC,WC",,"Recorder 4, channel 17"
```

3.5 INSTALLATION

3.5.1 Guidelines

GENERAL

- Profibus specified terminators (390Ω/220Ω/390Ω for Type A; 390Ω/150Ω/390Ω for Type B) must be used at each end of the link (resistors 0.25 W min.). Category 5 terminators available from the manufacturer (Part no. CI026529) should be used with 100Ω Category 5 cable.
- Cable types within a segment should not be mixed. Wherever possible use cable which complies with Profibus Standard EN50170.
- Keep stub lengths to a minimum. The total capacitance of all stubs in a network must not exceed 25pF (12Mbit sec); 200pF (1.5Mbit/sec) or 600pF (500kbit/sec).
- Always use the lowest data rate consistent with acceptable performance.
- All site installations must comply with the Profibus Installation Guidelines for Profibus-DP/FMS, available from the local Profibus National Organisation (PNO, Part no. 2.112).

IN CUBICLE

- 24awg, solid core, low loss Category 5 FTP cable with mutual capacitance <60pF/m. should be used. The minimum acceptable bend radius is 6mm. Cable idents, cable supports/cleats etc. must not be over tightened, or in anyway distort the cable, as the crushing of the outer jacket can affect the characteristic impedance of the cable and generate reflected signals.
- The RJ45 plugs must be designed to accept solid core cable. Suitable connectors are available from the instrument manufacturer (Part no. CI250449).
- The maximum total number of nodes, without repeater, is 18.
- Total cable runs must not exceed 30 metres (12Mbit/sec) or 70 metres (1.5Mbit/sec) without a repeater. Runs between nodes must be kept as short as possible.
- When terminating the cable, unshielded lengths must not exceed 30 mm. in length.
- At each end of each segment, a terminator (Part No. CI026529) must be used.
- Test nodes should be used sparingly and never more than one per segment.

3.5.2 Cubicle wiring

Most LIN products are designed to communicate with I/O controller units, located within one or more cubicles (figures 3.5.2a and 3.5.2b). For ease of connection, RJ45 connectors and Category 5 cable are used within the cubicle.

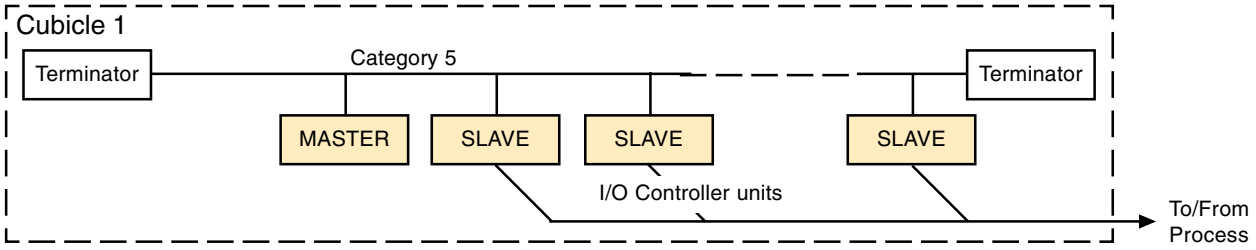
Note *For interfacing with external Profibus networks, use 9-way D-type connectors, and a suitable repeater (figure 3.5.2c). See Installation in the appropriate instrument handbook.*

The cable details given on this page refer to standard 150W Profibus cable. For Category 5 users, a document entitled 'Installation Guidelines for Profibus networks' is available from the manufacturer under part number HA261788.

Note *The Profibus link may need to be terminated using an appropriate connector, see Installation section in the appropriate instrument handbook.*

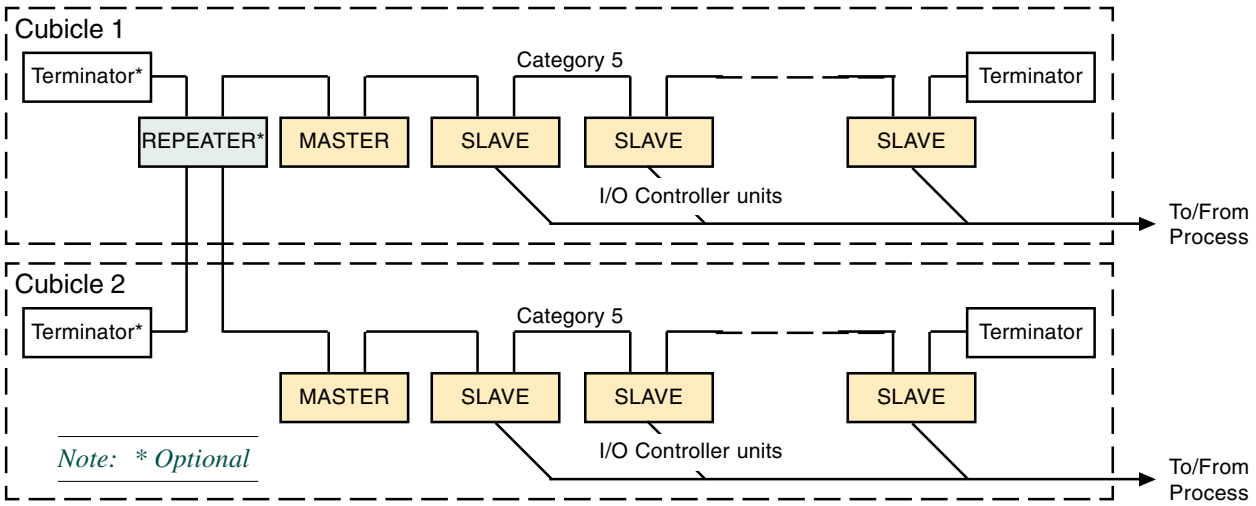
External profibus networks are normally implemented using Type A cable, see External Profibus Networks section, and are subject to the Profibus Standard guidelines available from the local Profibus National Organisation (PNO, Part no. 2.112).

3.5 INSTALLATION (Cont.)



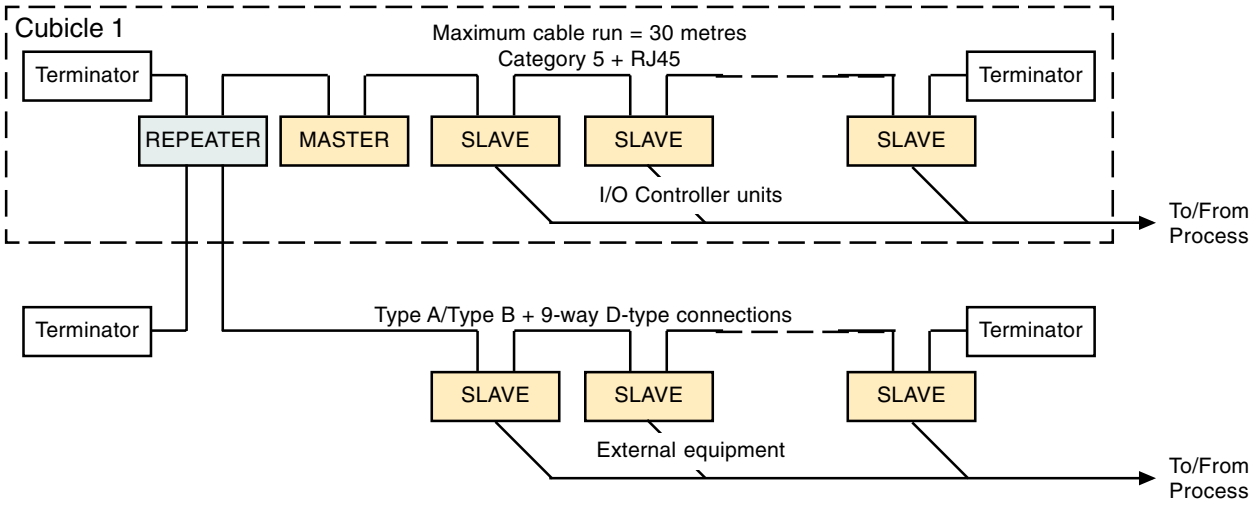
Note Maximum cable run = 30 metres. Maximum number of slaves = 31

Figure 3.5.2a single cubicle wiring



Note Maximum cable run = 30 metres per network. Maximum number of slaves = 61

Figure 3.5.2b Dual cubicle wiring



Note Maximum cable run = depends on installation and Baud Rate. Maximum number of slaves = 61

Figure 3.5.2c Single cubicle wiring with external equipment

3.5.3 External profibus networks

This section gives general guidelines only. For full details of Profibus installation, refer to Profibus Guidelines available from the local Profibus National Organisation (Part no. 2.142).

The cable details given below refer to standard 150Ω Profibus cable. Terminators must be Profibus approved for the cable type.

EARTHING THE SHIELD

The PROFIBUS standard suggests that both ends of the transmission line be connected to safety earth. If such a course is followed, care must be taken to ensure that differences in local earth potential do not allow circulating currents to flow, as these can not only induce large common mode signals in the data lines, resulting in communications failure, but can also produce potentially dangerous heating in the cable. Where doubt exists, it is recommended that the shield be earthed at only one point in each section of the network.

NETWORK WIRING

There are two distinct ways of wiring a network, known as 'Linear topology' and 'Tree topology'. In a linear network (figure 3.5.3a), the maximum number of repeaters is three, giving a total number of stations of 122. In theory the tree set-up (figure 3.5.3b) can have more stations, but the Profibus protocol limits the number of stations to 127 (addresses 0 to 126).

It is up to the user to determine which is the most cost effective way of organising the layout.

CABLE TYPE

Table 3.5.3a below gives the specification for a suitable Type A cable.

Cable parameter	Specification
Impedance	135 to 165 ohms at 3 to 20 MHz
Resistance	<110 ohms/km
Cable capacitance	< 30 pF/metre
Core diameter	0.34mm^2 max. (22awg)
Cable type	Twisted pair, 1x1, 2x2 or 4x1 lines
Signal attenuation	9db max. over total length of line section
Shielding	Cu shielding braid, or shielding braid and shielding foil

Table 3.5.3a Cable specification

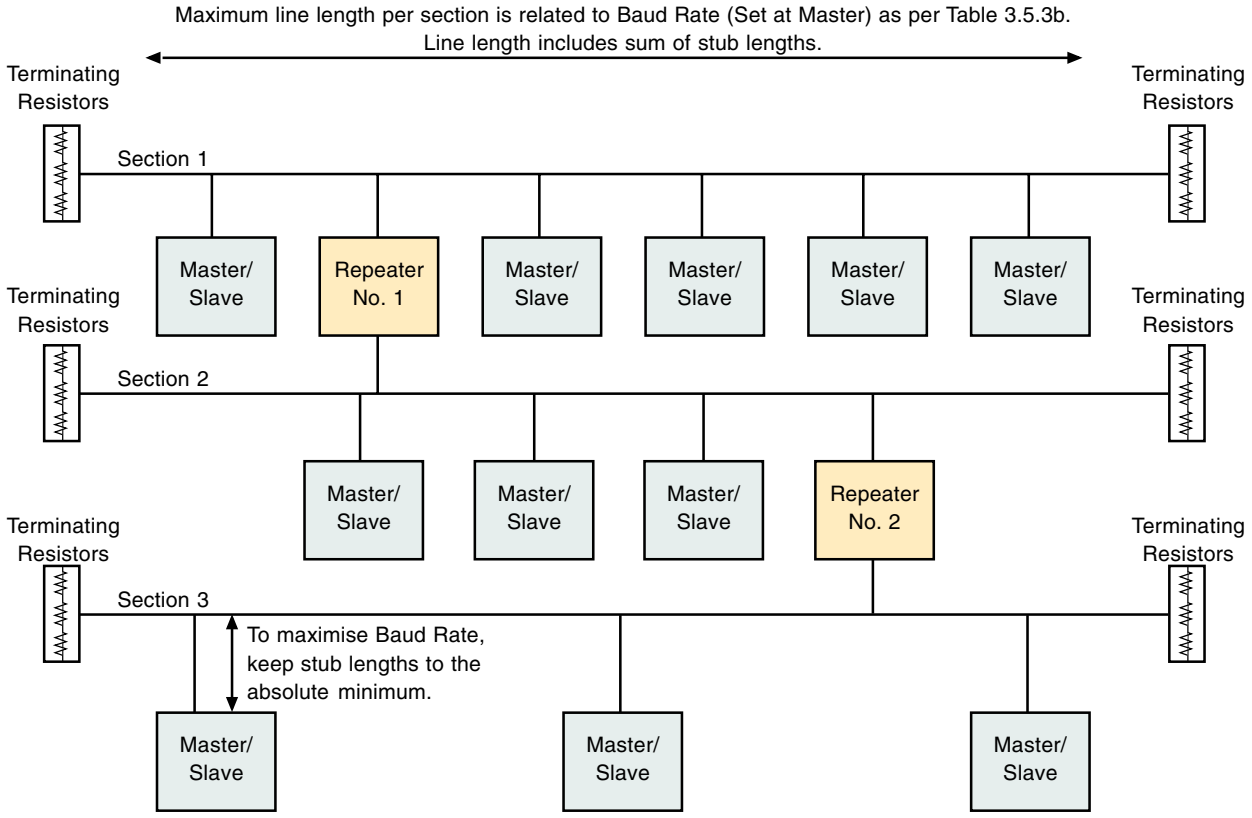
3.5 INSTALLATION (Cont.)

MAXIMUM TRANSMISSION RATE

The maximum transmission speed depends on the length of the cable run including 'stub' (distance from the bus to a station) lengths. Guaranteed minimum values for Type A cable (assuming maximum attenuation) are given in Table 3.5.3b, below. See Guidelines section for cable details.

Measure	Guaranteed value				
Line length/segment (metres)	100	200	400	1000	1200
Ma x Baud Rate (kbit/sec) (kB)	12,000	1,500	500	187.5	93.75

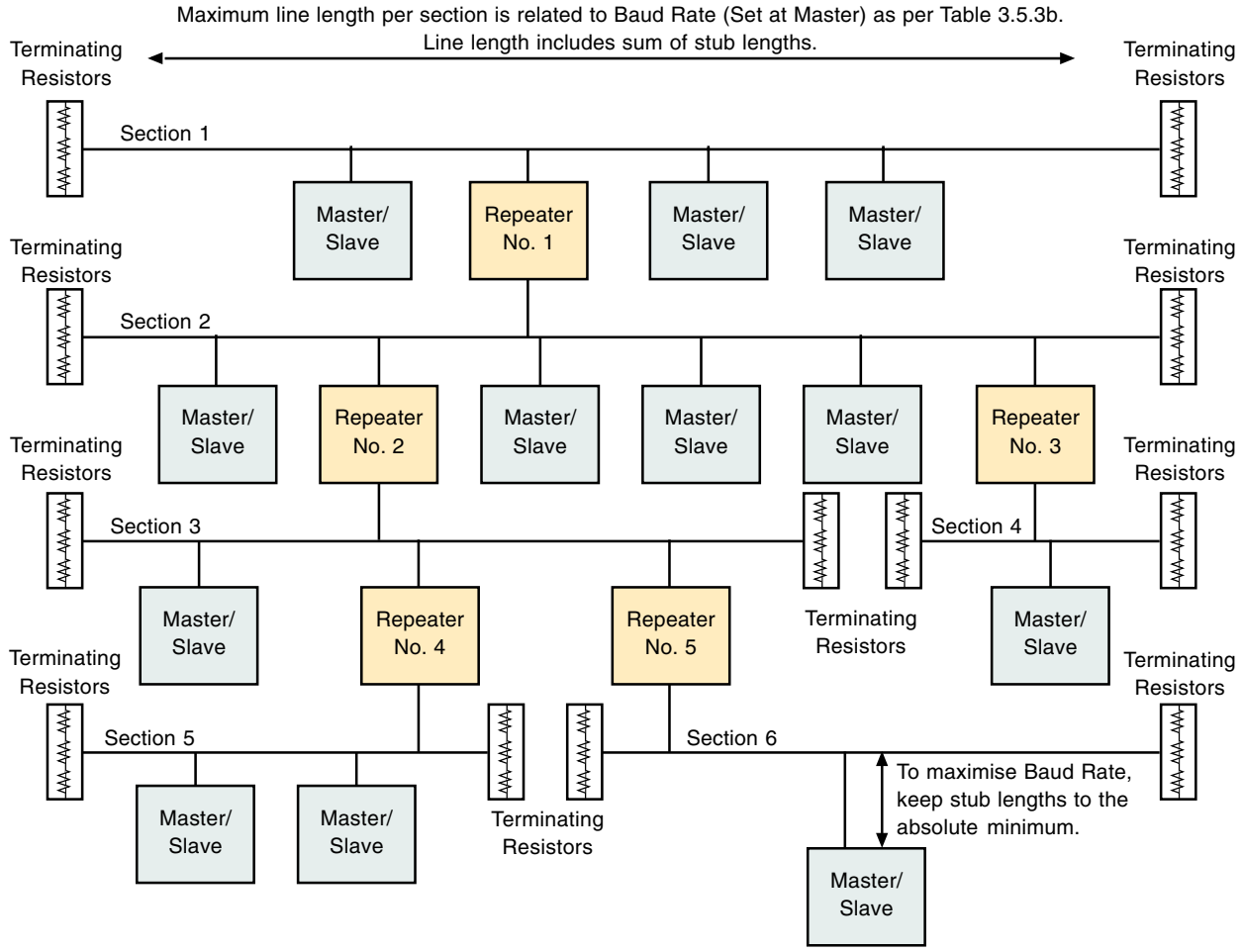
Table 3.5.3b Maximum Baud rate versus line length



Note *Typical linear bus layout, with 2 repeaters permitting a maximum of 14 slaves. A maximum of 3 repeaters is permitted, allowing up to 13 slaves.*

Figure 3.5.3a Typical linear bus layout

3.5 INSTALLATION (Cont.)



Note Typical tree bus layout, with 5 repeaters permitting a maximum of 11 slaves.

Figure 3.5.3b Typical tree bus layout

3.5.4 Adding a LIN product to the network

Some LIN products once they have been physically connected, required the hidden library, .uyp files and any .gsd files for third party equipment to be transferred, and the LIN Database started, see appropriate instrument handbook for Profibus details.

3.6 TROUBLESHOOTING

WARNING

Fault finding may affect the network and control system. Ensure that no damage to personnel or equipment can be caused by any fault finding activity.

NO COMMUNICATIONS

1. Check the wiring
2. Check the node address, ensuring that it is unique.
3. Ensure that the network has been correctly configured and that the configuration has been correctly transferred to the Master.
4. Verify that the .GSD file being used is correct, by loading it into the .gsd file configurator to check the format.
5. Ensure that the maximum line length of transmission line has not been exceeded for the Baud rate in use, see Table 3.5.3b.
6. Ensure that the final node on the transmission line (no matter what type of instrument it is) is terminated correctly using a terminator unit. Some equipment has built-in pull up and pull down resistors which in some cases can be switched in and out of circuit. Such resistors must be removed or switched out of circuit for all but the instruments at each end of the line.
7. Replace any faulty item(s) and re-test.

INTERMITTENT FAILURE TO COMMUNICATE

This fault is shown by the diagnostic status changing, without alarms being generated in the instrument. The following section details diagnostics information.

1. Check wiring as for 'No Communications' above. Pay particular attention to the integrity of the screening and termination.
2. Check the number of words in the data exchange against the maximum number the master can support.
3. Ensure that the maximum line length of transmission line has not been exceeded for the Baud rate in use, see Table 3.5.3b.
4. Ensure that the final node on the transmission line (no matter what type of instrument it is) is terminated correctly, and that only the first and final nodes are so terminated. Some equipment has built-in pull up and pull down resistors which in some cases can be switched in and out of circuit. Such resistors must be removed or switched out of circuit for all but the instruments at each end of the line.
5. Replace any faulty item(s) and re-test.

DATA FORMAT OR PARAMETER DATA SEEMS INCORRECT

Verify that the .gsd file is correct for the given application by loading it into a .GSD file configurator program.

COMMUNICATION SEEMS SLOW

The normal cyclic exchange of data should be very fast. Should so much data be requiring transfer that it cannot be fitted into the cycle rate, then it will be sent acyclically, and this results in a much slower transfer rate of all data.

To maximise efficiency, module DCMs should be used wherever possible instead of individual channel DCMs. Module DCMs provide a process variable's value and alarm status only.

Note *The amc_diag block provides information showing any communications 'overflow'. The pmc_diag block provides profibus diagnostics, see the LIN Blocks Reference Manual (Part no. HA 082 375 U003) for more details.*

3.7 GLOBAL COMMANDS

Freeze and Sync from a PROFIBUS master have no effect.

3.8 OPERATION

PROFIBUS DP performs a cyclical scan of the network devices, during which input and output data for each node is exchanged.

Values from each node (input data) are read by the Profibus controller, which then runs its control program, and generates a set of values (output data) to be transmitted to the nodes. This process is called an 'I/O data exchange'. This process is repeated continuously, to give a cyclical I/O data exchange.

Input data Examples:

- A set of digital readings for a digital input
- The measured temperature and alarm status from a PID controller.

Output data Examples:

- A setpoint to be sent to a PID controller

The I/O data exchange can be repeated continuously, can be synchronised at given times, or can be repeated at a pre-defined interval, which is asynchronous with the controller. Each node is normally assigned a group of PLC I/O registers, or a single function block, so that the controlling program can deal with each node's data as though the node is an internal device, without having to be concerned about timing problems. This mapping of node to register or function block is carried out during network configuration, which is usually carried out using a PC based program.

3.8.1 I/O data transfer limits

The PROFIBUS DP standard allows up to 244 bytes of data, or 116 discrete data items to be transferred in each direction, during each I/O data exchange. Many PLC masters, however, are unable to support more than 32 bytes, and this has become a typical value. Input and output data lengths for a given node are variable, and it is possible to define nodes as read only, write only or read/write.

The I/O data mixture used by a given slave device is defined by what is called a '.gsd' file, which can be edited to change the mapping of node parameters to PROFIBUS inputs and outputs. This file is imported into the network configuration before the network is created.

3.8.2 Data format

Data is transmitted in both directions as a single 16-bit integer value (also called a 'register'). The value is returned as a scaled integer such that 999.9 is returned as 9999, and 1.234 is returned as 1234. The control program in the PROFIBUS Master must convert these integers into floating point numbers if required. Alternatively, scaled integer number types can be used in the .uyp file to achieve the same end, see The .UYP File section.

3.9 THE .GSD FILE

Figure 3.9 shows that for each instrument on the communications link, a Device Database File is constructed and loaded into the Profibus configuration terminal. These files, called GeräteStammDaten or .GSD files, contain information, relating to the instrument's parameters, which the PROFIBUS master needs in order to communicate with the device.

When operating as a Profibus slave unit (not supported this version), it is necessary to load a .gsd file into the master unit before communications can be established. A suitable .gsd file is supplied with the unit.

When operating as a master unit, .gsd files are required for all 'third party' equipment with which the instrument is to communicate. Such .gsd files are normally supplied with the third party equipment. The instrument will normally come ready loaded with suitable .gsd files for I/O systems (for example) supplied by the instrument manufacturer.

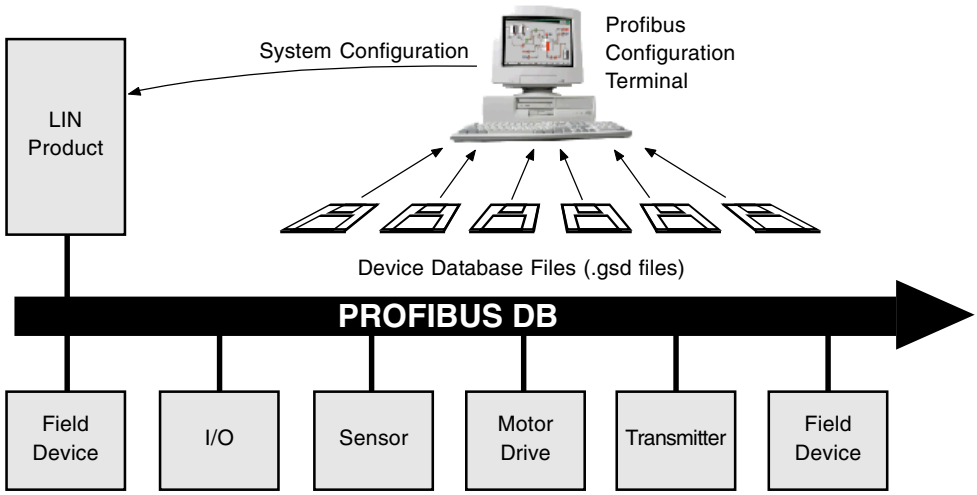


Figure 3.9 Typical PROFIBUS link using PLC as Master

(see http://www.profibus.com)

3.9.1 Demand data

If the LIN product has been configured to support demand data, this line must be added to the .gsd file that applies to that instrument.

Instrument	Keyword	Value	Description
Visual Supervisors (V3/2)			
	Eurotherm_Demand_Data	1	584SV, 590, 605, 690, etc.
		2	T630, 2500, T2550, etc.
		3	4103, 4100G.
	Eurotherm_Data_Control_Time	ms	Default value of 10 ms, see appropriate instrument specification.
	Eurotherm_TTR	Tbit	Override token rotation time
	Eurotherm_TSL	Tbit	Override slot time, see appropriate instrument specification.
	Eurotherm_TTD	Tbit	See Profibus specification.
	Eurotherm_G	1 to 100	See Profibus specification.
	Eurotherm_HAS	1 to 125	Override automatic HAS calculation, see appropriate instrument specification.
	Eurotherm_Max_Retry_Limit	0 to 7	See Profibus specification.

Table 3.9.1 Product Specific Keywords

3.10 REDUNDANT (DUPLEX) MODE

Specific LIN products permit redundant instrument processing to be carried out, so a failure in one processor will not affect control of the I/O units. Redundant wiring to the I/O units is not possible, however, any break in the network communications will cause the primary to lose control and the secondary to changeover, see appropriate instrument handbook for Profibus details.

3.10.1 Redundancy decisions

Some LIN Products can be configured as a duplex (redundant) pair. The primary and secondary units will independently derive the profibus status, and each will calculate a 'Profibus error weight'.

Normal redundant operation will take place only if the primary processor believes that both processors have an equal view of the Profibus slaves. If the 'Error weight' for the primary processor is higher than that of the secondary processor, the redundant pair will desynchronise. If both units have an error weight of 1, the units will changeover to try to achieve a better result.

The decision (to remain synchronised, desynchronise, decouple or changeover) is always made by the current primary processor, and then only whilst the redundant pair are synchronised.

Note An attempt to synchronise will be allowed to complete, and only after completion will the decision be made.

The decision is also deferred if the error weight is unstable. This prevents spurious desynchronise, decouple or changeover decisions being made as faults are introduced to or removed from the Profibus network.

APPENDIX A CONFIGURATION FILES

Some LIN Instruments contain files that define the configuration of the instrument, see appropriate instrument handbook. Changing specific configuration parameters of the I/O subsystem is achieved by editing files located in the LIN Instrument filesystem.

IMPORTANT It is recommended that the files are edited using the appropriate software tools. However, LIN products that do not support the use of the Instrument Properties dialog, and Process instruments that do not have on-screen menus, can be edited using an appropriate text editor, i.e. 'notepad.exe'.

- The _SYSTEM.OPT File (section A.1)

  These configuration options (_SYSTEM.OPT file) are specifically used for the instrument communications configuration, and include port, protocol and general communications parameter configuration.

  Note It is recommended that the Instrument on-screen menus or the Instrument Properties dialog is used to configure these parameters.

- The _SYSTEM.UXM File (section A.2)

  These configuration options (_SYSTEM.UXM file) are specifically used for Modbus TCP/IP configuration, and include the ports and runtime tuning parameters configuration.

  Note It is recommended that the Instrument on-screen menus or Modbus Tools is used to configure these parameters.

- The NETWORK.UNH File (section A.3)

  These configuration options (NETWORK.UNH file) are specifically used for LIN products, and include the IP, LIN and Setup configuration.

  Note It is recommended that the Instrument on-screen menus or the Instrument Properties dialog is used to configure these parameters.

A.1 THE _SYSTEM.OPT FILE

This file is specifically used to retain a set of instrument and communication parameters specific to each LIN product.

IMPORTANT All the information retained by this file should be edited by means of either on-screen menus or the Instrument Options tab on the Instrument Properties dialog, see Instrument Properties Help (Part no. RM 029 278) or Modbus Tools Help (Part no. HA 028 988). Using a text editor is not recommended.

Example:

This shows the instrument and communications parameters.

```
CONTRAST,5
LANGUAGE,0
URGENT_ALARM,OA
HOT,FALSE
WARM,TRUE
COLD,FALSE
DATE_FMT,0
FDATE_FMT,0
SDATE_FMT,0
TIME_FMT,0
FTIME_FMT,0
STIME_FMT,0
DURATION_FMT,0
KEYPAD_SIZE,0
HOME_TO,120
ENTRY_TO,10
POPUP_TO,10
SAVER_TO,0
ACCESS_TO,0,ACCESS_TO,0
COM1,RS485,None,0,9600,None,8,1,0,,Default
COM2,RS485,Panel820,0,38400,None,8,1,0,,Default
COM3,RS232,Modbus-S,3,9600,None,8,1,0,Default,,
COM4,RS232,None,0,9600,None,8,1,0,,Default
PROFIBUS1,RS485,None,0,1500000,None,99,99,250,
PROFIBUS2,RS485,None,0,1500000,None,99,99,250,
ALIN1,ARCNET,None,0
ENET1,ETHERNET,ELIN,8,0,None
ENET2,ETHERNET,FTP,0,0,None
ENET3,ETHERNET,Modbus-S,20,2000,COM1
ENET4,ETHERNET,Modbus-M,0,2000,None
ENET5,ETHERNET,Termcfg,0,0,
```


Note See Table for descriptions.

Figure A.1 _SYSTEM.OPT file parameters

Number	Function	Explanation
1	Port	Communications port name, COMn, Profibusn, ALINn, ENETn.
	Note	*To disable the protocol include the appropriate 'ENETn' entry and specify the protocol as 'None'.*
2	Communication Standard	Modbus **RS422** is 5-wire only. **RS485** can be 3-wire or 5-wire. Modbus DCM **RS422** is 5-wire only. **RS485** can be 3-wire or 5-wire. Profibus **RS485** can be 3-wire or 5-wire.
3	Protocol	Modbus - Instrument operating mode **Modbus-S** is Modbus Slave operating mode. **Modbus-TCP** is Modbus using TCP protocol operating mode Modbus DCM - Instrument operating mode. **Modbus-M** is Modbus Master operating mode. Profibus - Instrument operating mode **ProfibusDpv1-M** is Profibus Master operating mode.
4	Address	Modbus - Address of Instrument. **1 to 247** must be used for Instruments operating as Modbus Slave. **0** (zero) must be used for instruments operating via Modbus-TCP. Modbus DCM - Address of Instrument. **0** (zero) must be used for instruments operating via Modbus-TCP. Profibus - Address of Instrument operating **2** should be used for Instrument operating as an active master unit. **1** will then be automatically allocated to the passive master unit.
5	Baud Rate	Modbus - Value corresponding to Instrument configuration. **1200**, **2400**, **4800**, **9600**, **19200**, **38400** Modbus DCM - Value corresponding to Instrument configuration. **1200**, **2400**, **4800**, **9600**, **19200**, **38400** Profibus - Set automatically. **9600**, **19200**, **187500**, **500000**, **1500000**, **3000000**, **6000000**, **12000000**
6	Parity	Modbus - Value corresponding to Instrument configuration. **None**, **Odd**, **Even** Modbus DCM - Value corresponding to Instrument configuration. **None**, **Odd**, **Even** Profibus - Not Used.
7	Data Bits	Modbus - Value corresponding to Instrument configuration. **7** Not used. **8** Preconfigured Default. Modbus DCM - Value corresponding to Instrument configuration. **7** Not used. **8** Preconfigured Default. Profibus - Not Used.
8	Stop Bits	Modbus - Value corresponding to Instrument configuration. **1** Preconfigured Default. **2** User-defined, if required. Modbus DCM - Value corresponding to Instrument configuration. Profibus - Not Used.
9	Timeout in msec	Modbus - Time period until an error is indicated. **50 to 5000** must be used for Instruments operating as Modbus Master. **0** must be used for Instruments operating as Modbus Slave. Modbus DCM - Time period until an error is indicated. **50 to 5000** must be used for Instruments operating as Modbus Master. Profibus - Time period until an error is indicated. **0 to 5000** must be used for Instruments operating as Profibus Master.

continued...

Number	Function	Explanation
10	Transparent Modbus Access (TMA)	Modbus - Communications protocol for the instrument operating as a Master. *Note To configure TalkThru via Modbus TCP, the Address parameter must be 0 (zero),or the address of the Modbus Slave gateway. When this file is used on the T940X, port Com3 can be assigned to Modbus TCP communications.* **None**, indicates TMA is not required. **Modbus-M**, indicates TMA is using this instrument as the Modbus Master. **ProfibusDpv1-M**, indicates TMA is using this instrument as the Profibus Master. Modbus DCM - Communications protocol for the instrument operating as a Master. **None**, indicates TMA is not applicable. **Modbus-M**, indicates TMA is using this instrument as the Modbus Master. **ProfibusDpv1-M**, indicates TMA is using this instrument as the Profibus Master. Profibus - Communications protocol for the instrument operating as a Master. **None**, indicates TMA is not applicable. **Modbus-M**, indicates TMA is using this instrument as the Modbus Master. **ProfibusDpv1-M**, indicates TMA is using this instrument as the Profibus Master.
11	Three/five wire circuitry	Modbus - T940(X) only. Wiring circuitry used by the instrument. **Five**, indicates that EIA422 wire circuitry supports 5-wire only, and EIA485 wire circuitry supports 3-wire or 5-wire. **Three**, indicates that EIA485 wire circuitry supports 3-wire or 5-wire. Modbus DCM - Wiring circuitry used by the instrument. **Five**, indicates that EIA422 wire circuitry supports 5-wire only, and EIA485 wire circuitry supports 3-wire or 5-wire. **Three**, indicates that EIA485 wire circuitry supports 3-wire or 5-wire. Profibus - Wiring circuitry used by the instrument. **Five**, indicates that EIA422 wire circuitry supports 5-wire only.
12	Indirection Tables	Modbus - T800 Only. Register mapping in Slave instruments only. **None**, indicates that Indirection Tables are disabled. This is applicable to instrument using Modbus TCP, but ALL can be set if using a single master configuration. **RO**, indicates that Read-Only Indirection Tables are used. **RW**, indicates that Read-Write Indirection Tables are used. **All**, (Default) indicates that Read-Only and Read-Write Indirection Tables are used and should not be changed unless the tables are set up manually. Modbus DCM - Register mapping in Slave instruments only. Not applicable. Profibus - Register mapping in Slave instruments only. Not applicable.
13	Retries	Modbus - T800 Only. Communication attempts allowed before *Timeout* period expires and an error is indicated. **0 to 9**, (**2** Preconfigured Default) attempts allowed. Modbus DCM - Communication attempts allowed before *Timeout* period expires and an error is indicated. Not applicable. Profibus - Communication attempts allowed before *Timeout* period expires and an error is indicated. Not applicable.

Table A.1 _SYSTEM.OPT file parameter descriptions

A.2 THE _SYSTEM.UXM FILE

This file is specifically used to retain a set of Modbus TCP/IP communication parameters specific to each LIN product.

IMPORTANT All the information retained by this file can be edited by means of either on-screen menus or on pages via the TCP Properties dialog, in the Modbus Tools, see Modbus Tools Help (Part no. HA028988). Using a text editor is not recommended.

Example:

This example shows the specific communications parameters.

```
[Main]Port1=TCPIP
HostResolutionRetryFrequency=300000
ConnectInitialDelay=100
ConnectFailRetry1Frequency=1000
ConnectFailRetry2Frequency=2000
ConnectFailRetry3Frequency=5000
ReconnectRetryFrequency=0
ReconnectRetries=5
AsyncConnectPollTimeout=10
AsyncConnectTimeout=0
Name=TCPIP
[TCPIP]
Device1=T820_01
[MODBUS_1.DEV1]
InstrumentNr=0
CommsAddress=0
Port=502
Hostname=192.168.111.2
[MODBUS_1.T820_01]
InstrumentNr=1
CommsAddress=0
Port=502
Hostname=192.168.111.2
```

Figure A.2 _SYSTEM.UXM file

Field	Function
Main	
Port<n>=<PortName>	The *generic* port names, that are mapped to one of the enumerations in the Port field of a DCM function block. The generic ***Port<n>*** names are free format and must be followed by a ***[<PortName>]***.
HostResolutionRetryFrequency	The rate in milliseconds that the host name of a Modbus TCP instrument device will be resolved. Modbus TCP instrument devices can use TCP/IP host names or TCP/IP addresses, if addresses are used no resolution of the name is required. If this parameter is missing the 300000 default value applies.
Note	*Sometimes the Host name can be resloved by changing this value, but setting it too low may cause additional complications.*
ConnectInitialDelay	The ***initial*** delay incurred before attempting to connect to a Modbus TCP instrument device immediately its host name has been resolved in milliseconds. If this parameter is missing the 100 default value applies.
Note	*All connection requests are executed asynchronously to avoid blocking the Modbus communications in the* ***connect() call.***
ConnectFailRetry1Frequency	The delay incurred before the first retry to connect to a Modbus TCP instrument device in milliseconds. If this tuning parameter is missing a default value of 1000 will apply.
ConnectFailRetry2Frequency	The delay incurred before the second retry to connect to a Modbus TCP instrument device in milliseconds. If this parameter is missing the 2000 default value applies.
ConnectFailRetry3Frequency	The delay incurred before the third retry to connect to a Modbus TCP instrument device in milliseconds. If this parameter is missing the 5000 default value applies.
ReconnectRetryFrequency	The delay incurred between subsequent retries (after third) to connect to a Modbus TCP instrument device in milliseconds. If this parameter is missing the 0 default value applies.
ReconnectRetries	The number of retries to connect to a Modbus TCP instrument device at the ***ReconnectRetryFrequency***. If this parameter is missing the 5 default value applies. If the number of retries is complete, it returns to ***ConnectFailRetry1Frequency*** and repeats.
AsyncConnectPollTimeout	The time that Modbus communications will wait for a connection to succeed or fail whilst polling for the completion of a connection request in milliseconds. If this parameter is missing the 0 default value applies. This defines the amount of time the Modbus communications will hang in the ***select()*** call.
AsyncConnectTimeout	The time in milliseconds that Modbus communications will accept before cancelling a connection request that has yet to complete. If this parameter is missing the 0 default value applies. The 0 value relies on the Windows implementation of the ***connect()*** call to fail a connection request, any other value is used to override the time imposed by Windows.
TCPIP	
PortIsTCPIP	The revision level. Only values 2 or greater are accepted.
Name=<ModbusPort>	The current implementation, <*ModbusPort*> to be MODBUS_1.
Enabled	The Modbus port status. If zero the port is disabled.
ModbusTCP	The Modbus communications is using ModbusTCP and not Serial Modbus.
Device<x>=<DeviceName>	ModbusTCP instrument device names, where ***<x>*** can be any combination of alphanumeric characters. Typically devices will be named using Device1, Device2, … etc. For each named device there should be a section named ***[<ModbusPort>.<DeviceName>]***.
MODBUS PORTP	
CommsAddress	The decimal Modbus address of the ModbusTCP instrument device.
HostName=<Name>	The TCP/IP host name of the ModbusTCP instrument device or the TCP/IP address in standard decimal notation. If a host name is used the Modbus communications will need to resolve it.
Port	The decimal TCP/IP port number for the Modbus communications in the ModbusTCP instrument device. Is this parameter is missing 502 will be used by default.
InstrumentNr	The decimal Modbus address as defined in the GW tables ‘Instr No’.

Table A.2 _SYSTEM.UXM file parameters

A.3 THE NETWORK.UNH FILE

This file is specifically used to retain a set of network parameters specific to each LIN product.

IMPORTANT The network.unh is not only used by LIN instruments, but also by LINOPC on the PC for configuring the manufacturers Network Explorer.

Note Always use the on-screen menus, or the Instrument Properties dialog to configure these parameters. Although not recommended, the Terminal Configurator connected via a Telnet session, can be used to edit these parameters.Detailed information about this file is described in the ELIN User Guide (Part no. HA082429)

The network parameters, retained by this file, display the same editable settings as accessed via the LINOPC Applet available from the PC Control Panel directory.

- IP

  These settings define how the IP configuration will be allocated.

- LIN

  These settings define the LIN protocol configuration.

Example:

When despatched from the factory, the 'network.unh' file enables DHCP with Link-Local Fallback:

```
# This section is for EuroPRP

[LIN]
#Defines the LIN protocol name - default if not explicitly defined is "NET"
#ProtocolName = NET

[IP]
DHCP=on
LinkLocal=on
#IPaddress=149.121.165.23
#Subnet=255.255.255.0
```

Figure A.3 Default 'network.unh' file

Field	Function
LIN	
ProtocolName=NET	'NET' is the default Network Port Name and should be changed to ensure address conflicts between instruments do not occur.
Note The fields displayed in the LIN section do not apply to Modbus communications.	
IP	
DHCP=on	'On' indicates that the Dynamic Host Configuration Protocol (DHCP) is enabled. 'Off' shows it is disabled. DHCP is a superset of the IP parameter acquisition part of BootP.
Note Instruments that do not support DHCP will use BootP in lieu of DHCP.	
LinkLocal=on	'On' indicates that a local network-unique IP address is derived by negotiating with the other IP hosts on the same network. 'Off' shows it is disabled. Operates as a fallback from DHCP, and to BootP. Link-Local can also be used in its own right as the primary method of IP address allocation.
IPaddress	Shows the network-unique IP address of the instrument.
Subnet	Shows the A subnetwork-unique address of the instrument allowing local network of IP hosts to connect and communicate directly to one other without the use of a Gateway.

Table A.3 Network.unh file parameters

RECOVERY FROM UNKNOWN IP ADDRESS CONFIGURATION

LIN products differ in the way an IP address is configured. If an IP address should need to be recovered, the process of recovery will differ between LIN products, see appropriate instrument handbook.

Note *Always use the on-screen menus, or the Instrument Properties dialog to configure these parameters. Although not recommended, the Terminal Configurator connected via a Telnet session, can also be used to edit these parameters.*

Index

COMMUNICATIONS MANUAL

Symbols

.cnf file A-1
.unh file A-1, A-7
  Unknown IP Address A-8
.uym file 2-1, 2-3
.uyp file 3-4
  Examples 3-7
_system.opt file 2-2, 3-1, 3-2, A-1
_system.uxm file A-1
16-bit 1-8
  CNOMO registers 1-8
32-bit 1-8
  Long signed 1-8
  Long unsigned 1-8
  Swapped 1-9

A

Adding unit to the Profibus network 3-14
Address 1-13
  0 to 15 1-10
  16 to n 1-10
amc_diag block 3-15

B

BAT_CTRL block 1-9
Baud Rate 3-13
Block
  See also Function block

C

Cabling 3-12
Cache block 1-1, 1-2, 1-9
Channel value 2-2, 2-3
CNOMO 1-2
  Registers 1-2, 1-8
Commenting 2-5, 3-9
Configuration Tools 1-2, 1-3, A-7, A-8
  iTools 1-2, 1-3
  LINtools 1-2
  Modbus Tools 1-2
  Terminal Configurator A-7, A-8
    Telnet A-7, A-8
Cyclic data structure 3-7

D

Data coherence 1-9
Data conversion 1-8
  Digital 1-8
  Register 1-8
Data format 2-4, 3-16
  DINT 2-4
  DINT_X 2-4
  INT 2-4
  REAL 2-4
  REAL_X 2-4
  SINT 2-4
  SREAL_p1 2-4
  SREAL_p2 2-4
  SREAL_p3 2-4
  SREAL_p4 2-4
  STIME_dh 2-4
  STIME_dm 2-4
  STIME_ds 2-4
  SUREAL_p1 2-4
  SUREAL_p2 2-4
  SUREAL_p3 2-4
  SUREAL_p4 2-4
  TIME 2-4
  UDINT 2-4
  UDINT_X 2-4
  UINT 2-4
  USINT 2-4
Date and Time 1-9
  ISO8601 format 1-9
  POSIX format 1-9
DCM block 1-9, 3-3
Diagnostic 1-10, 1-12
  Function codes 1-12
  Registers 1-10
  Table 1-10
Diagnostic Function Codes 1-12
Diagnostic registers 1-10
  0 to 15 1-10
  16 to n 1-10
  Address 1-13
  Internal 1-10
  Table status and control 1-10
DINT Data format 2-4
DINT_X Data format 2-4

E

Ethernet 1-3
Exception responses 1-12

F

Floating-point numbers 1-8
Function block 1-1, 1-4, 1-9, 2-1, 3-4, 3-15, 3-16, A-6

G

GW_CON block 1-2
GW_TBL block 1-2

I

Instrument Properties 2-2, 3-2
INT Data format 2-4
IP Address A-1, A-8
ISO8601 format 1-9
iTools 1-2, 1-3

L

LIN 1-1, A-1
  Database 1-1, 1-10
  Product 1-5, 1-10, 1-12, 3-1
  Unknown IP Address A-8
LIN Blocks
  See also Function Block
Linear Bus layout 3-13
LINOPC A-7
LINtools 1-2, 2-2, 3-2
Long signed 32-bit 1-8
Long unsigned 32-bit 1-8

M

Master mode 1-5, 1-10
Memory use and requirements 1-7
  Digital image data 1-7
Modbus 1-1
  Address 1-13
  Control registers 1-10
  Data formats 2-4
  DCM 2-1
    .uym file 2-1, 2-3
    Commenting 2-5
  Diagnostic function codes 1-12
  Diagnostic tables 1-10
  Exception responses 1-12
  Gateway 1-1
  Implementation 1-13
  Serial 1-1
  Table status 1-10
  TCP 1-1, A-1, A-5, A-6
    Communication parameters A-5
  Value scaling 2-5
Modbus Tools 1-2
Mode
  Master device 1-5, 1-10
  Slave device 1-5, 1-10
Module identifier 3-5

N

Network wiring 3-12
network.unh file A-1
  Unknown IP Address A-8

O

Offset parameter 1-4
  0 to 15 1-10
  16 to n 1-10
Operating mode 1-5, 1-10

P

Parameter Setup 2-1
  Modbus 2-2, 3-1, 3-2
PID controller 3-16
PLC 3-1
pmc_diag block 3-15
Polling period 1-5
POSIX format 1-9
Profibus 3-1
  .gsd files 3-17
  Adding unit to network 3-14
  Cabling 3-12
  Commenting 3-9
  Cyclic Read/Write data structure 3-7
  Data Format 3-16
  DCM 3-3
  I/O data transfer limits 3-16
  Linear Bus layout 3-13
  Memory image 3-7
  Network wiring 3-12
  Operation 3-16
  Redundancy decisions 3-18
  Troubleshooting 3-15
  Wiring guidelines 3-10
Programmable Logic Controller 3-1

R

Real Data format 2-4
Real-Time Clock 1-9
REAL_X Data format 2-4
Redundancy
  Decisions 3-18

S

SCADA 1-3
Scaling 2-5
Scanner task 1-2
Sequence 1-11
SINT Data format 2-4
Slave mode 1-10
SPP_CTRL block 1-9
SPP_RAMP block 1-9
SREAL_p1 Data format 2-4
SREAL_p2 Data format 2-4
SREAL_p3 Data Format 2-4
SREAL_p4 Data format 2-4
STIME_dh Data format 2-4
STIME_dm Data format 2-4
STIME_ds Data format 2-4
Supervisory Control And Data Acquisition 1-3
SUREAL_p1 Data format 2-4
SUREAL_p2 Data format 2-4
SUREAL_p3 Data Format 2-4
SUREAL_p4 Data format 2-4

T

Table Status and Control Registers 1-10
TalkThru 1-2, 1-3
Terminal Configurator A-7, A-8
TIME Data format 2-4
TIMEDATE block 1-9
Timing information 1-6
TMA 1-2, 1-3
TOT_CONN block 1-9
TOTAL block 1-9
Transparent Modbus Access 1-2, 1-3
Troubleshooting 3-15

U

UDINT block 2-4
UDINT_X block 2-4
UINT block 2-4
USINT block 2-4

V

Value
  16-bit integer 2-3
  Alarm threshold 2-2, 2-3
  Analogue input 2-5
  Channel High range 2-5
  Channel Low range 2-5
  Scaled 2-5

W

Wiring 3-10
  Profibus guidelines 3-10

Intentionally left blank

Inter-Company sales and service locations

AUSTRALIA Sydney
Eurotherm Pty. Ltd.
Telephone (+61 2) 9838 0099
Fax (+61 2) 9838 9288
E-mail info.au@eurotherm.com

AUSTRIA Vienna
Eurotherm GmbH
Telephone (+43 1) 7987601
Fax (+43 1) 7987605
E-mail info.at@eurotherm.com

BELGIUM & LUXEMBURG Moha
Eurotherm S.A/N.V.
Telephone (+32) 85 274080
Fax (+32) 85 274081
E-mail info.be@eurotherm.com

BRAZIL Campinas-SP
Eurotherm Ltda.
Telephone (+5519) 3707 5333
Fax (+5519) 3707 5345
E-mail info.br@eurotherm.com

DENMARK Copenhagen
Eurotherm Danmark AS
Telephone (+45 70) 234670
Fax (+45 70) 234660
E-mail info.dk@eurotherm.com

FINLAND Abo
Eurotherm Finland
Telephone (+358) 22506030
Fax (+358) 22503201
E-mail info.fi@eurotherm.com

FRANCE Lyon
Eurotherm Automation SA
Telephone (+33 478) 664500
Fax (+33 478) 352490
E-mail info.fr@eurotherm.com

GERMANY Limburg
Eurotherm Deutschland GmbH
Telephone (+49 6431) 2980
Fax (+49 6431) 298119
E-mail info.de@eurotherm.com

HONG KONG & CHINA
Eurotherm Limited North Point
Telephone (+85 2) 28733826
Fax (+85 2) 28700148
E-mail info.hk@eurotherm.com

Guangzhou Office
Telephone (+86 20) 8755 5099
Fax (+86 20) 8755 5831
E-mail info.cn@eurotherm.com

Beijing Office
Telephone (+86 10) 6567 8506
Fax (+86 10) 6567 8509
E-mail info.cn@eurotherm.com

Shanghai Office
Telephone (+86 21) 6145 1188
Fax (+86 21) 6145 1187
E-mail info.cn@eurotherm.com

INDIA Chennai
Eurotherm India Limited
Telephone (+9144) 2496 1129
Fax (+9144) 2496 1831
E-mail info.in@eurotherm.com

IRELAND Dublin
Eurotherm Ireland Limited
Telephone (+353 1) 4691800
Fax (+353 1) 4691300
E-mail info.ie@eurotherm.com

ITALY Como
Eurotherm S.r.l
Telephone (+39 31) 975111
Fax (+39 31) 977512
E-mail info.it@eurotherm.com

KOREA Seoul
Eurotherm Korea Limited
Telephone (+82 31) 2738507
Fax (+82 31) 2738508
E-mail info.kr@eurotherm.com

NETHERLANDS Alphen a/d Rijn
Eurotherm B.V.
Telephone (+31 172) 411752
Fax (+31 172) 417260
E-mail info.nl@eurotherm.com

NORWAY Oslo
Eurotherm A/S
Telephone (+47 67) 592170
Fax (+47 67) 118301
E-mail info.no@eurotherm.com

POLAND Katowice
Invensys Eurotherm Sp z o.o
Telephone (+48 32) 218 5100
Fax (+48 32) 217 7171
E-mail info.pl@eurotherm.com

SPAIN Madrid
Eurotherm España SA
Telephone (+34 91) 661 6001
Fax (+34 91) 661 9093
E-mail info.es@eurotherm.com

SWEDEN Malmo
Eurotherm AB
Telephone (+46 40) 384500
Fax (+46 40) 384545
E-mail info.se@eurotherm.com

SWITZERLAND Wollerau
Eurotherm Produkte (Schweiz) AG
Telephone (+41 44) 787 1040
Fax (+41 44) 787 1044
E-mail info.ch@eurotherm.com

UNITED KINGDOM Worthing
Eurotherm Limited
Telephone (+44 1903) 268500
Fax (+44 1903) 265982
E-mail info.uk@eurotherm.com
Web www.eurotherm.co.uk

U.S.A Leesburg VA
Eurotherm Inc.
Telephone (+1 703) 443 0000
Fax (+1 703) 669 1300
E-mail info.us@eurotherm.com
Web www.eurotherm.com

ED52

invensys
EUROTHERM
EUROTHERM LIMITED
Faraday Close, Durrington, Worthing, West Sussex, BN13 3PL
Telephone: +44 (0)1903 268500 Facsimile: +44 (0)1903 265982
e-mail: info.uk@eurotherm.com
Website: http://www.eurotherm.co.uk

Specification subject to change without notice. ©Eurotherm Limited.